Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)

29 November 2012

Lisbon, Portugal

Editors:

Francesco Mambrini

Marco Passarotti

Caroline Sporleder

Title: Proceedings of the Second Workshop
on Annotation of Corpora for Research in the Humanities
(ACRH-2)

Editors: Francesco Mambrini, Marco Passarotti,
Caroline Sporleder

Cover photograph: Pedro Salitre

ISBN 978-989-689-273-9

Depósito legal n.º 351 423/12

Publisher: Edições Colibri, Lisboa
www.edi-colibri.pt

Sponsors:

Faculdade de Letras
da Universidade de Lisboa

Lisbon, November 2012

Preface

Research in the Humanities is predominantly text-based. For centuries scholars have studied documents such as historical manuscripts, literary works, legal contracts, diaries of important personalities, old tax records etc. Manual analysis of such documents is still the dominant research paradigm in the Humanities. However, with the advent of the digital age this is increasingly complemented by approaches that utilise digital resources. More and more corpora are made available in digital form (theatrical plays, contemporary novels, critical literature, literary reviews etc.). This has a potentially profound impact on how research is conducted in the Humanities. Digitised sources can be searched more easily than traditional, paper-based sources, allowing scholars to analyse texts quicker and more systematically. Moreover, digital data can also be (semi-)automatically mined: important facts, trends and interdependencies can be detected, complex statistics can be calculated and the results can be visualised and presented to the scholars, who can then delve further into the data for verification and deeper analysis. Digitisation encourages empirical research, opening the road for completely new research paradigms that exploit `big data' for humanities research. This has also given rise to Digital Humanities (or E-Humanities) as a new research area.

Digitisation is only a first step, however. In their raw form, electronic corpora are of limited use to humanities researchers. The true potential of such resources is only unlocked if corpora are enriched with different layers of linguistic annotation (ranging from morphology to semantics). While corpus annotation can build on a long tradition in (corpus) linguistics and computational linguistics, corpus and computational linguistics on the one side and the Humanities on the other side have grown apart over the past decades. We believe that a tighter collaboration between people working in the Humanities and the research community involved in developing annotated corpora is now needed because, while annotating a corpus from scratch still remains a labor-intensive and time-consuming task, today this is simplified by intensively exploiting prior experience in the field. Indeed, such a collaboration is still quite far from being achieved, as a gap still holds between computational linguists (who sometimes do not involve humanists in

developing and exploiting annotated corpora for the Humanities) and humanists (who sometimes just ignore that such corpora do exist and that automatic methods and standards to build them are today available).
ACRH-2 aims to foster communication and collaboration between these two groups, in the same way that its predecessor ACRH-12 did. ACRH-12 was held at Heidelberg University on January 5, 2012, in conjunction with the 10th edition of the international workshop on "Treebanks and Linguistic Theories" (TLT-10). ACRH-2 is again co-located with TLT, this time at the University of Lisbon. We received thirteen submissions for ACRH-2. After a thorough reviewing process eight submissions were included in the workshop, addressing several important issues related to corpus annotation for the Humanities.

The papers in the proceedings concern several different topics. The task of resource creation is tackled by Koeva et al., who present an aligned parallel Bulgarian-English corpus for linguistic research, and Ferreira et al., who introduce a novel framework for annotating corpora with a particular focus on language documentation. Four papers are concerned with corpora of historical texts which pose particular challenges for language processing software. A major problem are spelling variations. Detecting and normalising these is addressed by two papers: Bollmann test several string distance methods for Early New High German, while Reynaert et al. compare two state-of-the-art error detection systems on old Portuguese. Historical texts also often lack consistent punctuation, which poses difficulties for automatic segmentation into linguistic units. The paper by Petran presents a method for segmenting texts that lack punctuation marks into sentences, clauses and chunks. In turn, Bouma and Hermans introduce an algorithm for syllabification in Middle Dutch text. Finally, two papers are concerned with deeper processing problems. Both focus on folktale corpora. Everhardus et al. present an approach for normalisation and consistency checking in semi-structured corpora, while Karsdorp et al. address the task of identifying actors and ranking them by importance. The workshop programme is completed by an invited lecture by Martin Wynne, who heads the Oxford Text Archive and has worked extensively in the areas of corpus linguistics and corpus infrastructure development.

We are grateful to everybody who made this event possible, including Erhard Hinrichs, the local and non-local organisers of TLT-11 (in particular Iris Hendrickx), the ACRH-2 programme committee, and the authors who submitted papers. We also acknowledge the endorsement of the AMICUS network.

The ACRH-2 Co-Chairs and Organisers
Francesco Mambrini (University of Cologne, Germany)
Marco Passarotti (Università Cattolica del Sacro Cuore, Milan, Italy)
Caroline Sporleder (Trier University, Germany)

Program Committee

Chairs:
Francesco Mambrini (University of Cologne, Germany)
Marco Passarotti (Università Cattolica del Sacro Cuore, Milan, Italy)
Caroline Sporleder (Trier University, Germany)

Members:
David Bamman (USA)
Gabriel Bodard (UK)
Lars Borin (Sweden)
Antonio Branco (Portugal)
Milena Dobreva (Malta)
Anette Frank (Germany)
Dag Haug (Norway)
Erhard Hinrichs (Germany)
Beáta Megyesi (Sweden)
Petya Osenova (Bulgaria)
Martin Reynaert (the Netherlands)
Victoria Rosén (Norway)
Jeff Rydberg Cox (USA)
Melissa Terras (UK)
Manfred Thaller (Germany)
Martin Volk (Switzerland)

Organising Committee

Chairs:
Francesco Mambrini (University of Cologne, Germany)
Marco Passarotti (Università Cattolica del Sacro Cuore, Milan, Italy)
Caroline Sporleder (Trier University, Germany)

Local Committee:
Amalia Mendes
Iris Hendrickx
Sandra Antunes
Aida Cardoso
Sandra Pereira
all University of Lisbon, CLUL, Portugal

Contents

Do we need annotated corpora in the era of the data deluge?

Martin Wynne
University of Oxford, UK
E-mail: martin.wynne@oucs.ox.ac.uk

Abstract

Language corpora were originally developed as datasets for linguistic research, in a world where researchers rarely had access to machine-readable language data. Corpus linguistics subsequently developed methodologies based on discrete, bounded datasets, created to represent certain types of language use, and studied as exemplars of that domain. The growth of the field and advances in technology meant that corpora became bigger and more plentiful and various, with huge reference corpora for a vast range of languages and time periods, and numerous specialist corpora. Researchers in many other fields have found that corpora are rich repositories of data about not only language but also culture, society, politics, etc.. Annotation has been widely used, initially by linguists, but also by researchers in other fields to categorize, interpretate and analyse texts, and to aid information retrieval and data linking.

Nowadays, the enormous wealth of digital language data at our fingertips brings the role of the corpus into question. Born-digital data, along with large-scale digitization of historical texts, are delivering the cultural products of the both the present and the past directly to our desktops. We can relatively easily make bespoke datasets for different research questions. The boundaries between the corpus and other type of data are becoming blurred. Can we still justify spending our time carefully crafting and annotating corpora today?

There is a danger that adding annotation is too time-consuming and costly, and forces us into using smaller and older datasets, and only starting our research after they have been painstakingly annotated. Furthermore, there is a lack of generic software which can make use of annotations, and so we build separate web interfaces for each annotated corpus. This leads to the now-familiar problem of the creation of digital silos - isolated from other corpora and tools, limited in functionality, and each with a different interface. Thus it can be argued that adding annotation to a corpus also adds to the problem of fragmentation of digital resources in the humanities.

These new difficulties come along at a time when we continue to wrestle with longstanding problems with annotation:

- How do we avoid the circularity of adding annotations and then counting and analysing them ourselves - “finding the Easter eggs that you hid in the garden yourself”?
- Will two humanists ever agree on the relevance or usefulness of any given annotation scheme, or the accuracy of any instantiation of it?
- Can automatic annotation be accurate enough to be useful to the humanities scholar?
- Do we risk ignoring the actual data itself by focussing on our interpretations and annotations?

The era of the data deluge poses additional questions. Do we now need to explore how far can we go in our research without carefuly crafted, annotated corpora? Or do we need better and faster tools to add automatic annotations to the deluge of data? Or will initiatives such as CLARIN make it easier to use a wide range of annotated corpora on common platforms?

(Semi-)Automatic Normalization of Historical Texts using Distance Measures and the Norma tool

Marcel Bollmann

Department of Linguistics
Ruhr-Universität Bochum
E-mail: bollmann@linguistics.rub.de

Abstract

Historical texts typically show a high degree of variance in spelling. Normalization of variant word forms to their modern spellings can greatly benefit further processing of the data, e.g., POS tagging or lemmatization. This paper compares several approaches to normalization with a focus on methods based on string distance measures and evaluates them on two different types of historical texts. Furthermore, the Norma tool is introduced, an interactive normalization tool which is flexibly adaptable to different varieties of historical language data. It is shown that a combination of normalization methods produces the best results, achieving an accuracy between 74% and 94% depending on the type of text.

1 Introduction[1]

Historical language data poses a difficult challenge for natural language processing. One of the biggest problems is the lack of standardized writing conventions: the exact characters and symbols used to spell a word can depend on many different factors such as dialectal influences, space constraints on a page, or personal preferences of the writer. Consequently, while annotation tasks such as part-of-speech (POS) tagging are well explored for modern languages, with reported accuracies as high as 97% (Brants [5]), the same cannot be said for historical varieties. In a study done on Early New High German (Scheible et al. [12]), tagging accuracies using taggers trained on modern German were below 70%. A possible solution is a pre-processing step during which word forms are converted to their modern spellings, which the same study found to increase tagging accuracy by 10 percentage points. This process will be referred to here as 'normalization'.

[1]The research reported here was financed by Deutsche Forschungsgemeinschaft (DFG), Grant DI 1558/4-1.

When creating a corpus of historical texts, manually normalizing all data can be a tedious and time-consuming process. Furthermore, despite the many inconsistencies, spelling is typically not arbitrary, and many regularities can still be found. In Early New High German (ENHG), typical examples are *'v'* for (modern) *'u'*, as in *vnd – und* 'and', or *'y'* for (modern) *'i'*, as in *seyn – sein* 'be'. This leads to the question of how the process of normalization can be automated. Several methods for automatic normalization have been proposed (some of the more recent ones are, e.g., Baron et al. [3], Jurish [9], Bollmann et al. [4]), though no clear "best" method has been identified. On the contrary, it is unclear to which extent the results are even comparable, as the amount and degree of spelling variance can be expected to differ greatly between texts. At the same time, this variance suggests the need for normalization methods which are flexibly adaptable to different types of texts.

This paper addresses these issues in two ways: (1) by comparing different normalization methods that can be flexibly trained to handle different spelling varieties; and (2) by presenting the interactive normalization tool Norma, which provides a framework to combine, train, and compare different methods of normalization.

The structure of this paper is as follows. Sec. 2 presents a rule-based method for normalization, while Sec. 3 describes a normalization approach based on string distance measures. Sec. 4 introduces the Norma tool. In Sec. 5, all presented normalization methods are evaluated on two different types of historical texts. Sec. 6 discussed the comparable VARD tool, Sec. 7 presents related work and further conceivable approaches to the normalization task, and Sec. 8 concludes.

2 Rule-Based Method

Rule-based normalization will be used as a comparison to the distance measures described in Sec. 3. Its main concepts are only briefly summarized here; a detailed description can be found in Bollmann et al. [4].

The rule-based method applies a set of character rewrite rules to a historical input string in order to produce its normalized variant. Rewrite rules operate on one or more characters and take their immediate context into account; an example rule is shown in Ex. (1).

(1) `j → ih / # _ r`
('j' is replaced by 'ih' between a word boundary ('#') and 'r')

The rules are not meant to be specified manually, but are supposed to be learned from training data (using a modified Levenshtein algorithm). To normalize an input string, it is processed from left to right, with one rewrite rule being applied at each position. Typically, there will be several applicable rules at each given position; in this case, the frequency of a rule during training determines how likely it is to be used during the normalization process. In case there is no applicable rule, the

default is to leave the character at that position unchanged. Additionally, to prevent the generation of nonsense words, all normalization candidates are matched against a target lexicon. Therefore, it is possible that the rule-based method sometimes fails to find any normalization candidate at all.

3 Distance Measures

Levenshtein distance between two strings is defined as the number of substitutions, insertions, and deletions required to transform one string into the other (Levenshtein [11]). More generally, the term "distance measure" will be used here to describe any function that accepts a pair of strings and returns a positive, numeric value describing their distance.

Any distance measure can theoretically be used for the purpose of normalization. Given a (finite) lexicon of modern word forms, the distance of a historical input word form to each entry in the modern lexicon can be calculated; the word form with the lowest distance to the input string is then chosen as the normalization.[2]

3.1 FlexMetric

FlexMetric (Kempken [10]) is a measure originally designed for rating historical and dialectal spelling variants, which makes it appear ideal for the normalization task. It is supposed to be flexible in the sense that it can easily be adapted to different types of text and language varieties. This is achieved by using a variant of Levenshtein distance with weights, where instead of simply counting the number of edit operations, each such operation is assigned a weight w (also called "cost") with $0 < w \leq 1$. Weights can be individually defined; if $w = 1$ for all edit operations, this approach is identical to standard Levenshtein distance.

However, instead of allowing arbitrary weights, FlexMetric introduces the concept of character groups, which contain characters that are closely related to each other. Conceivable groupings for German are, e.g., *{s, z, ß}* or *{i, y}*. These groups are then assigned individual costs, which in turn define the cost of character substitutions within that same group. Substitutions of characters that do not belong to the same group are assigned a default cost of 1. Note that this approach induces a symmetry: the substitution $i \rightarrow y$ always costs the same as $y \rightarrow i$.

Additionally, in the extended version of the algorithm, which is used here, multiple occurrences of the same character are ignored—e.g., the transformation $n \rightarrow nn$ (and vice versa) always has a cost of 0.

The relative simplicity of this approach is deliberate; the intention was to allow for the creation of a good parametrization by manual inspection of a small portion of the input data and without too much effort (Kempken [10], p. 61). Again, this seems well-suited for normalization, as historical data typically shows a lot

[2] In practice, finding an efficient algorithm for this operation can be a limiting factor, though.

of variety between texts, requiring an easily adaptable normalization method, and training data is typically not available (or not in large amounts).

3.2 MultiWLD

In the Luther text used for evaluation (cf. Sec. 5), the substitution $v \rightarrow u$ is by far the most common one; however, the opposite direction, $u \rightarrow v$, is comparatively rare. This leads to the question whether costs should actually be defined in a symmetric way. Furthermore, it is questionable whether characters should be treated in isolation: e.g., *'ck'* often occurs for modern *'k'*, as in *werck – werk* 'work', and does not represent a different phoneme. Treating this example simply as a deletion of *'c'* would be a case of overgeneralization, though. Many similar examples for German can be found ($mb \rightarrow m$, $i \rightarrow ie$, $a \rightarrow ah$, etc.).

These considerations led to the MultiWLD[3] algorithm. It is supposed to be similar to FlexMetric in the sense that costs should be easily definable by intuition and/or manual inspection of a small input sample, but differs in the aspects mentioned above: (1) it is directional (instead of symmetric); and (2) it allows the definition of substitutions with more than one character on either side.

Although these changes draw nearer to the idea of the rule-based approach (cf. Sec. 2), there are two important differences to keep in mind. Firstly, the distance measures described here only assign a cost to actual modifications to the input string; i.e., it is always preferred that a character is left unchanged, which is not the case in the rule-based method. Secondly, while the definition of multi-character substitutions introduces a form of context-sensitivity (e.g., $a \rightarrow ah$ represents the insertion of *'h'* after *'a'*), it is up to the creator of the parametrisation how much and which context, if any, to include in each rule. In contrast, the rule-based approach always requires exactly one character on each side as context.

4 The Norma tool

Norma is a tool for (semi-)automatic normalization of historical language data. It can be used both interactively and non-interactively, and is intended to be flexible in the sense that it is not restricted to any particular method(s) of normalization. Normalization methods—or "normalizers"—are encapsulated in external modules for easy extensibility. Parameter sets of normalizers can be dynamically retrained during the normalization process if the normalizer supports it. This way, Norma supports an incremental learning approach to normalization.

Norma is written in Python 2.7.[4] At the time of writing, only a command-line interface is available.

[3] MultiWLD = Weighted Levenshtein Distance with multi-character substitutions

[4] `http://www.python.org/`

4.1 Modes of operation

Norma supports three modes of operation: batch, interactive, and training mode.

In batch mode, Norma takes an input file consisting of one historical word form per line and, for each line, outputs the suggested normalization of that word form.

Interactive mode works the same way as batch mode, but prompts the user with each generated normalization, which he or she can either confirm or correct. The resulting pair of historic and (confirmed) modern word form is then used to train the normalizers. Ideally, the number of corrections the user has to make should decrease over time as the normalizers are supplied with more training data and can adjust their set of parameters accordingly.

Finally, there is a training mode, which works similarly to interactive mode except that the confirmed modern word form is not entered by the user, but given in the input file itself. Apart from training normalizers from a set of manually normalized data, this mode can also be used to "simulate" the interactive normalization process, e.g. to evaluate the effect of incremental learning.

4.2 Normalizer modules

Norma itself does not implement any normalization or learning techniques, but rather depends on external normalizer modules to provide that functionality.

Normalizers exchange data with Norma via a certain, pre-defined interface of functions. The normalization function is given a historical word form as input and should return its normalized form along with a confidence score. Confidence scores are numerical values between 0 and 1; a score of 0 is taken to mean that no suitable normalization could be found at all. The training function is given a historical word form along with its modern counterpart and is expected to trigger the normalizer's training algorithm.

4.3 Normalization cycle

The normalization process in Norma can be described as a cycle consisting of the following steps:

1. fetching a historical input word form;
2. generating a normalization candidate for the historical word form using one or more normalizers;
3. validating the candidate word form against the correct normalization; and
4. training the normalizers with the historical word form and its correct normalization.

This cycle is repeated until all input word forms have been processed. If training is disabled or impossible, e.g. when batch-processing texts, the last two steps will do nothing. Fig. 1 shows a flow diagram visualizing the normalization cycle.

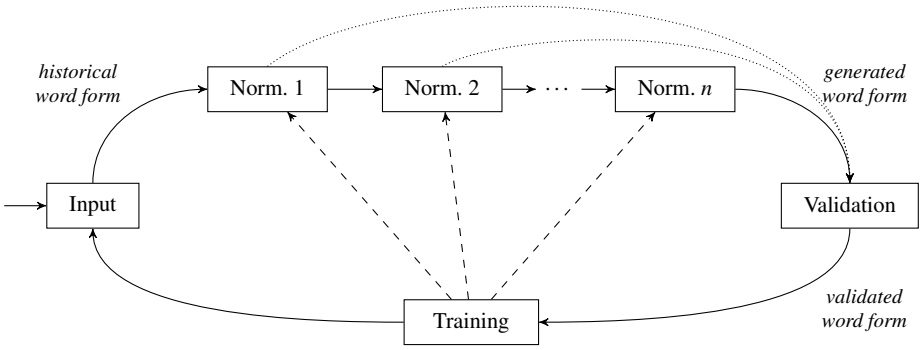


Figure 1: Flow diagram of a normalization cycle (with training) in Norma

The first step is the retrieving of a historical word form as input. Typically, this will be read from a text file, from which lines are processed in sequential order.

The second step in the cycle is the actual normalization process. It is modelled as a chain: at the beginning, the first normalizer in the chain is called and supplied with the historical word form; if it finds a suitable normalization with a confidence score above a pre-defined threshold, the chain is interrupted and processing continues with the next step (indicated in Fig. 1 by the dotted-line arrows). Otherwise, the next normalizer in the chain is called. If all normalizers fail to produce an acceptable result, the unmodified original word form is used as the normalization candidate.

Afterwards, the normalization candidate is validated against the correct normalization. In interactive mode, the user is prompted to either confirm the candidate form or correct it, while in training mode, the correct form is read directly from the input file.

As the last step of the cycle, the training function of all normalizers is called, with the historic word form and its (confirmed) modern equivalent as input. After the normalizers have been trained, the cycle starts anew.

5 Evaluation

5.1 Corpora

For the evaluation, two different texts from ENHG were used: (1) the Luther bible in its version from 1545 and a revised modern version of it; and (2) a manuscript of the Anselm corpus from the Eastern Upper Germany dialectal area[5] that was manually normalized by a student assistant. A more detailed description is given in Bollmann et al. [4].

[5]The depository of the manuscript is Melk, Austria.

		Luther		Anselm	
Training	Total tokens	218,504	—	500	—
Evaluation	Total tokens	109,258	(100.00%)	4,020	(100.00%)
	Identical tokens	71,163	(65.13%)	1,512	(37.61%)
	Maximum accuracy	103,409	(94.65%)	3,760	(93.53%)

Table 1: General corpus statistics, split up by training and evaluation part. "Identical tokens" are counted between the original text and its modern version; "maximum accuracy" gives the highest accuracy that can be achieved with context-free normalization methods.

Table 1 shows some general corpus statistics. The Luther bible is a comparatively large text, with 109,258 tokens in the evaluation part, and is relatively close to its modern version: 65% of all tokens in the historical version are already identical to their modern counterparts. This figure serves as the baseline for the evaluation. The Anselm text, on the other hand, is much smaller (4,020 tokens) and shows a lot more variance in spelling. While the size of the Luther text—and the training part in particular—makes it more suitable for training automatic normalization methods, the Anselm text is closer to actual research scenarios: texts with a high variance in spelling, and without a reasonable amount of training data available (if any).

It is also interesting to note the maximum achievable accuracy, which is around 94% for both texts. All methods evaluated in this paper are context-free, i.e., they operate on word forms in isolation. Consequently, source tokens of the same type will be assigned identical normalizations. However, the same type can be aligned with more than one target token: a typical example is the historical *jn* mapped to either *in* 'in' or *ihn* 'him', depending on context. In this case, it is impossible to achieve perfect accuracy using context-free methods. Therefore, the "maximum accuracy" in Table 1 represents the accuracy that would be achieved if each type was normalized to the token it is most often aligned with.

5.2 Evaluation procedure

Five normalization methods were evaluated: rule-based normalization (Sec. 2), standard Levenshtein distance (Sec. 3), FlexMetric (Sec. 3.1), MultiWLD (Sec. 3.2), and a wordlist mapper. The mapper simply learns word-to-word substitutions from training data without any notion of character distance. If a word form can be mapped to multiple modern word forms, the mapping with the highest frequency is used; in case of a tie, the mapping that was learned first is chosen.

Parametrization for the rule-based method and the wordlist mapper has been learned from separate training parts of the texts as shown in Table 1. For FlexMetric and MultiWLD, parametrization was created by hand after inspection of at most

500 tokens from the training part of each text. As an example, characters groups learned from Luther that were assigned the lowest costs are *{u, v}* and *{i, j}*, while the most common insertions/deletion were of *'e'* and *'h'*. This method is, of course, a highly subjective one and should only be seen as a first experiment on whether a reasonable parametrization can be created by manual inspection of a small text fragment alone.

For the target lexicon, I used the complete vocabulary of the modern Luther bible complemented by a full-form lexicon of all simplices generated by the German morphology DMOR (Schiller [13]).

In addition to the separate evaluation of each method, each possible chain combination (cf. p.) has been evaluated. Threshold for confidence scores was set to 0, i.e., unless the normalizer failed to find a normalization, any generated normalization candidate was accepted as the final result. As a consequence, chains in this test setup can consist of three elements at maximum, as the distance-based measures can never fail to find a normalization candidate.

All methods have been implemented as normalizer modules for Norma (cf. Sec. 4.2). Evaluation was done using Norma's training mode to produce parameter files for the rule-based method and the wordlist mapper, and batch mode to process the evaluation texts.

5.3 Results

Evaluation results are shown in Table 2. For the Luther text, the simple wordlist mapper performs best in isolation, achieving an accuracy of 92.60%, with the rule-based approach and MultiWLD being only slightly worse. This is a notable result for the adaptable (or "trained") distance measures in particular, as their parametrization was compiled manually and with much less training data. Combining the approaches yields an even better performance, up to a maximum of 93.72% when the mapper is used first, the rule-based approach second, and either MultiWLD or FlexMetric last. However, any chain that starts with the wordlist mapper results in a comparable score, making it hard to determine a single "best" combination here. It should be noted that these scores are very close to the maximum accuracy of 94.65%, showing that these approaches—and the chain configurations in particular—are almost optimal for the Luther text.

Results for the Anselm text are considerably worse than for Luther; again, this is to be expected, as the baseline is significantly lower (37.61%). Additionally, much less training data was used for the mapper and the rule-based method. Their scores fall below those of the trained distance measures, with MultiWLD now achieving the best accuracy (67.91%) and FlexMetric following close behind. This suggests that trained distance measures outperform both wordlist mappings and the rule-based approach when only a small amount of training data is available.

Combining methods results in an even bigger increase in accuracy with the Anselm text. The best result (73.63%) is achieved by using only the mapper and MultiWLD, although the configuration that was best for the Luther text also

	Luther		Anselm	
Baseline	71,163	(65.13%)	1,512	(37.61%)
Mapper	**101,170**	**(92.60%)**	2,448	(60.90%)
Rule-based	98,767	(90.40%)	2,530	(62.94%)
MultiWLD	96,510	(88.33%)	**2,730**	**(67.91%)**
FlexMetric	92,353	(84.53%)	2,655	(66.04%)
Levenshtein	87,619	(80.19%)	2,161	(53.76%)
Mapper → *Rule-based* → *MultiWLD*	**102,193**	**(93.53%)**	2,947	(73.31%)
Mapper → *Rule-based* → *FlexMetric*	**102,193**	**(93.53%)**	2,947	(73.31%)
Mapper → *Rule-based* → *Levenshtein*	102,160	(93.50%)	2,793	(69.48%)
Mapper → *MultiWLD*	102,131	(93.48%)	**2,960**	**(73.63%)**
Mapper → *FlexMetric*	102,118	(93.47%)	2,911	(72.41%)
Mapper → *Levenshtein*	101,867	(93.24%)	2,705	(67.29%)
Rule-based → *Mapper* → *MultiWLD*	98,890	(90.51%)	2,829	(70.37%)
Rule-based → *Mapper* → *FlexMetric*	98,890	(90.51%)	2,800	(69.25%)
Rule-based → *Mapper* → *Levenshtein*	98,857	(90.48%)	2,675	(66.54%)
Rule-based → *MultiWLD*	98,890	(90.51%)	2,815	(70.02%)
Rule-based → *FlexMetric*	98,890	(90.51%)	2,785	(69.28%)
Rule-based → *Levenshtein*	98,857	(90.48%)	2,660	(66.17%)

Table 2: Accuracies after normalization, with methods evaluated both separately and combined as a chain. "Baseline" gives the accuracy before normalization for comparison. Best results are highlighted in bold.

shows a similar performance here. Again, a clear "winner" cannot be determined. However, the tendency that the wordlist mapper should come first in the chain holds equally for the Anselm text as for Luther, despite the comparatively bad performance of the mapper on Anselm when used in isolation.

Untrained Levenshtein distance always performs worst, which is especially apparent in the separate evaluation. This suggests that it should probably never be used for the normalization task, or at least not in isolation, as a list of edit weights compiled from even a small input sample gives significant boosts in accuracy.

6 Comparison to VARD 2

A well known interactive tool for normalization of historical texts is VARD 2 (Baron and Rayson [2]). It was designed for use with Early Modern English texts and also combines different normalization methods. VARD is written in Java and is freely available for academic research.[6]

In contrast to Norma, VARD features a graphical user interface for interactive processing of texts. When a text is loaded into the tool, historical spelling variants are automatically detected and highlighted. Right-clicking on a variant word form

[6] `http://www.comp.lancs.ac.uk/~barona/vard2/`

shows a list of suggested normalization candidates; the user can choose to either normalize all occurrences of a word form or one particular instance only. VARD clearly offers much more convenience here. However, its approach is not functionally different, i.e., the same kind of user interface and functions could also be built on top of Norma.

The biggest difference between the tools, however, lies in the customizability of the normalization methods. VARD combines four different methods: 'known variants', which is similar to the wordlist mapper described in Sec. 5.2; 'letter replacement', which applies a list of letter replacement rules to a word; 'phonetic matching', which uses a modified Soundex algorithm to find modern word forms that are phonetically similar to the input word; and standard edit distance. Additionally, all candidate word forms are matched against a modern lexicon.

This set of normalization methods is fixed and can be neither reduced nor extended. Also, while some of these components can be modified, the phonetic matching algorithm is hard-coded to the phonetics of English and can neither be changed nor completely turned off. Consequently, when working with texts in other languages, this algorithm is most likely to produce incorrect results. Norma, on the other hand, places no restrictions on the use and combination of normalizers, which should enable an easier adaption to different languages.

Nevertheless, there has been work on adapting VARD to other languages, such as Portuguese (Hendrickx and Marquilhas [8]). To be able to compare VARD's performance with Norma's, a similar adaption has been tried for the German data. To this end, VARD has been given the same modern lexicon and the same letter replacement rules as the MultiWLD algorithm in the former evaluation (cf. Sec. 5.2). The training part of each text was then used to run VARD's training mode. Finally, batch mode with a threshold of 0%[7] was used to normalize the evaluation parts.

Final accuracies were 91.19% (Luther) and 69.38% (Anselm), both worse than the best scores in the evaluation with Norma (cf. Table 2). However, VARD performed better on the Anselm text than any normalization method in isolation. Interestingly, it performed worse on Luther than the wordlist mapper alone, despite having learned the same mappings from the training text.

7 Related Work and Outlook

Previous work on historical spelling variation tends to focus on the task of information retrieval (IR), which is basically reverse to normalization: finding historical variants that correspond to a modern input string (e.g., Baron et al. [3], Ernst-Gerlach and Fuhr [6], Hauser and Schulz [7]). It is unclear how well these approaches translate to the normalization task with a view to further processing of the data by NLP tools such as POS taggers.

[7] The threshold is the minimum F-score required for a normalization candidate to be used; a setting of 0% ensures that all variant word forms will be normalized, which is closest to the behaviour of Norma. Also, the default setting of 50% was found to perform significantly worse.

For future work, more distance measures could be included in the evaluation. Zobel and Dart [14] test different methods in the context of spelling correction and name matching, and find an n-gram string distance to be effective. Jurish [9] uses token context information and a combination of different methods via a hidden Markov model to normalize historical German, achieving high performance, though also evaluated as an IR task. Context information has not yet been considered for Norma, but is a possible line of future research.

Incremental learning is a core feature of Norma, but has so far been neglected in the evaluation. Parameter sets for FlexMetric and MultiWLD have been compiled manually with good results; it remains to be tested whether similar results can be achieved when learning parameters automatically from training data, and if so, how much training data is needed. Adesam et al. [1] describe a pilot study using automatically extracted replacement rules with promising results.

Finally, normalization also bears similarities to machine translation, in which words (to be normalized) correspond to sentences (to be translated) and characters correspond to words. This is especially apparent in the rule-based method (cf. Sec. 2) where character identities must be learned in the same way as substitutions, and where context information is taken into account, similar to the implementations of statistical translation models.

8 Conclusion

In this paper, several approaches to normalization were discussed and evaluated against two different historical texts. When used in isolation, the rule-based method and a simple wordlist mapper performed best on the Luther text, where a large amount of training data was available. Trained distance measures performed best on the Anselm text, where only a small amount of training data was used and which showed considerably more variation in spelling. In all cases, a combination of these methods proved to be best, achieving an accuracy of up to 93.72% for Luther and 73.63% for Anselm. In particular, the results showed that integrating a simple word-to-word mapper always increased accuracy.

Evaluation was performed using Norma, an interactive normalization tool which is easily extensible and more flexible than the comparable VARD tool. Norma was used here for its capability of employing, training, and combining different normalization methods, though the possibility of semi-automatic normalization using incremental learning techniques remains to be explored further.

References

[1] Yvonne Adesam, Malin Ahlberg, and Gerlof Bouma. *bokstaffua, bokstaffwa, bokstafwa, bokstaua, bokstawa...* Towards lexical link-up for a corpus of Old Swedish. In *Proceedings of KONVENS 2012 (LThist 2012 workshop)*, pages 365–369, Vienna, Austria, 2012.

[2] Alistair Baron and Paul Rayson. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*, 2008.

[3] Alistair Baron, Paul Rayson, and Dawn Archer. Automatic standardization of spelling for historical text mining. In *Proceedings of Digital Humanities 2009*, Maryland, USA, 2009.

[4] Marcel Bollmann, Florian Petran, and Stefanie Dipper. Applying rule-based normalization to different types of historical texts — an evaluation. In *Proceedings of LTC 2011*, pages 339–344, Poznan, Poland, 2011.

[5] Thorsten Brants. TnT — a statistical part-of-speech tagger. In *Proceedings of ANLP 2000*, pages 224–231, Seattle, USA, 2000.

[6] Andrea Ernst-Gerlach and Norbert Fuhr. Generating search term variants for text collections with historic spellings. In *Proceedings of the 28th European Conference on Information Retrieval Research (ECIR 2006)*. Springer, 2006.

[7] Andreas W. Hauser and Klaus U. Schulz. Unsupervised learning of edit distance weights for retrieving historical spelling variations. In *Proceedings of FSTAS 2007*, pages 1–6, Borovets, Bulgaria, 2007.

[8] Iris Hendrickx and Rita Marquilhas. From old texts to modern spellings: an experiment in automatic normalisation. *JLCL*, 26(2):65–76, 2011.

[9] Bryan Jurish. More than words: using token context to improve canonicalization of historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39, 2010.

[10] Sebastian Kempken. *Bewertung historischer und regionaler Schreibvarianten mit Hilfe von Abstandsmaßen*. Diploma thesis, Universität Duisburg-Essen, 2005.

[11] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[12] Silke Scheible, Richard J. Whitt, Martin Durrell, and Paul Bennett. Evaluating an 'off-the-shelf' POS-tagger on Early Modern German text. In *Proceedings of LaTeCH-2011*, pages 19–23, Portland, USA, 2011.

[13] Anne Schiller. Deutsche Flexions- und Kompositionsmorphologie mit PC-KIMMO. In Roland Hausser, editor, *Proceedings of the first Morpholympics*, Tübingen, 1996. Niemeyer.

[14] Justin Zobel and Philip Dart. Finding approximate matches in large lexicons. *Software: Practice and Experience*, 25(3):331–345, 1995.

Poio API - An annotation framework to bridge Language Documentation and Natural Language Processing

Peter Bouda, Vera Ferreira, António Lopes

Centro Interdisciplinar de Documentação Linguística e Social
Rua Dr. António Ferreira da Silva Totta 29, 2395-182 Minde, Portugal
E-mail: pbouda@cidles.eu, vferreira@cidles.eu, alopes@cidles.eu

Abstract

After 20 years of multimedia data collection from endangered languages and consequent creation of extensive corpora with large amounts of annotated linguistic data, a new trend in Language Documentation is now observable. It can be described as a shift from data collection and qualitative language analysis to quantitative language comparison based on the data previously collected. However, the heterogeneous annotation types and formats in the corpora hinder the application of new developed computational methods in their analysis. A standardized representation is needed. Poio API, a scientific software library written in Python and based on Linguistic Annotation Framework, fulfills this need and establishes the bridge between Language Documentation and Natural Language Processing (NLP). Hence, it represents an innovative approach which will open up new options in interdisciplinary collaborative linguistic research. This paper offers a contextualization of Poio API in the framework of current linguistic and NLP research as well as a description of its development.

1 Introduction

Language Documentation is a new and promising domain in linguistics. Through the data collected in several documentation projects during the last 20 years, a basis was created for systematic quantitative language comparison. However, to achieve this goal, a standardized representation of the existing data must first be created. This is what we intend to do with Poio API, a scientific software library written in Python.

After a brief introduction to Language Documentation (Part 2) and a short presentation of Natural Language Processing (Part 3) and Quantitative Language Comparison (Part 4), we will concentrate on the description of Poio API in the last section of this paper (Part 5).

2 Language Documentation

Language diversity, its documentation, and analysis have always interested linguists around the world, especially those working on language typology. However, the beginning of language documentation as it is known today is normally set during the last decade of the 20th century. Several factors contributed to the emergence of this "new" linguistic discipline. First of all, technological developments which enabled the recording, processing, and storage of large amounts of linguistic data with high quality portable devices and fewer storage necessities (i.e. by more efficient codecs) opened up new perspectives and possibilities for the work in the field, in and with the language communities. On the other hand, the interest in linguistic diversity and more specifically in endangered languages spread beyond the academic world and became a public issue, mainly through the continuous reports on the subject (some of them very populistic and without scientific foundation) published by the press and well-known institutions, such as the UNESCO with its Atlas of World's Languages in Danger[1]. This mediatization also contributed to the rise of financial support for the documentation and research of undocumented or poorly documented languages[2]. Additionally, the need to standardize the study and documentation of endangered languages became a current subject in academic discussions.

In this context, documentary linguistics ([8]) imposed itself with the aim of developing a "lasting, multipurpose record of a language" ([9]). The collection, distribution, and preservation of primary data of a variety of communicative events ([8]), i.e. real situations of language use in several contexts, emphasizes the difference between documentary linguistics and descriptive linguistics. In this sense, primary data include not only notes (elicited or not) taken by linguists during the work with the language community, but also, and above all, audio and video recordings, as well as photos and text collections. The data is normally transcribed, translated, and it should also be annotated. This task requires linguistic annotations (morpho-syntactic, semantic, pragmatic, and/or phonetic annotations,) as well as a broad range of non-linguistic annotations (anthropological, sociolinguistic, musical, gestual, etc. annotations) whenever possible and/or if important to the language community being documented. Even if no full annotation is made in the way described before (mostly because it is not manageable in the limited timespan of language documentation projects and/or the financial resources available do not permit to build real interdisciplinary teams), the fact of making primary data availabe presents the advantage that researchers from the same or from other dis-

[1] http://www.unesco.org/culture/languages-atlas/, accessed 30.8.2012

[2] See for instance the DoBeS program financed by the Volkswagen Foundation - http://www.mpi.nl/DOBES/dobesprogramme/, accessed 30.8.2012, The Hans Rausing Endangered Languages Project from SOAS in London - http://www.hrelp.org/, accessed 30.08.2012, or the program Documenting Endangered Languages from the National Science Foundation - http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=12816, accessed 30.08.2012, to refer only a few.

ciplines can use the data for their own purpose and complement it with their own annotations.

Typical end products of language documentation projects are:

- Multimedia corpora (with audio, video, photos, and annotations) properly archived;
- Dictionaries (frequently multimedia dictionaries);
- Sketch grammars of the documented language where the main characteristics of its grammatical system are described and which serve as a kind of user manual for the created corpus. The data included in the grammar should be entirely extracted from the collected data.

This new perspective on collecting, analyzing and distributing linguistic data brought by documentary linguistics has proven to be a very important step towards interdisciplinary research in Humanities and towards the improvement of accountability of linguistic research results.

However, several technical requirements must be fulfilled in order to ensure a "lasting, multipurpose" documentation of a language. As for data processing, two different softwares for transcriptions and annotations are widely accepted: ELAN, developed by the technical team at Max-Planck-Institute (MPI) in Nijmegen, and Toolbox, developed by SIL International. And, most important of all, the data must be archived and made available to researchers, language communities, and the general public.

Two of the best-known archives today that preserve and publish documentation on endangered languages are The Language Archive (particularly the DoBeS session in archive)[3] at MPI in Nijmegen and the Endangered Languages Archive (ELAR)[4] at SOAS in London.

3 Natural Language Processing

As mentioned in section 2, data from language documentation projects has always been used in analysis tasks. Researchers have written dictionaries, typological sketches or reference grammars about "their" language, based on the data they collected in the field. The digitization of a whole research field for data processing and archival purposes recently led to new types of quantitative studies emerging within the research fields of language typology, language classification and historical linguistics (see for example [16], [18]). This shift from qualitative to quantative analysis is now also observable in recent research with data from language documentation: digital archives provide corpora that are extensive enough to be used with established and new mathematical methods to process natural language.

[3] http://www.mpi.nl/DOBES/archive_info/, accessed 30.8.2012

[4] http://elar.soas.ac.uk/, accessed 30.8.2012

Natural Language Processing (NLP) is best understood in its widest sense, "any kind of computer manipulation of natural language" ([4]). It has become an integral part of computer-human-interaction and, as such, of people's everyday life all over the world. The start of NLP was closely related to the field of Artificial Intelligence (AI) and connected to research which aimed at understanding and simulating the human mind. A new approach based on statistics and stochastics in the 1980s was found superior to the classical AI systems ([17]). Although NLP has had the advantage of a vast financial support, most of the research has been concerned with systems that process major languages such as English, German, Spanish, etc., which are spoken by many potential end users in the economic centres of the 20th century. As these languages represent only a small part of the global linguistic diversity and are furthermore restricted to a sub-part of one language family, namely Germanic, Romanic, and occasionally Slavic languages from the Indo-European language family, most systems are highly limited when it comes to processing language in a broader sense. This becomes apparent when processing "new" big languages like Chinese and Arabic, and this has led to many developments and rapid progress in this research field.

4 Quantitative Language Comparison

Within the broad field of NLP the methods from corpus linguistics have been the first to be applied to the data from language documentation. A central task in most language documentation projects remains the annotation of the corpus. Thus, (semi-)automatic taggers based on statistical or rule-based tagging mechanisms developed in corpus linguistics support fieldwork and later analysis. But corpus linguistic methods have also been used to gain insights into how the languages work, something which is not anymore possible through human processing alone as soon as a researcher works on an extensive corpus of language data. Statistical models support the work of the linguist by showing regularities or deviations within large data sets. It soon became clear, though, that the existing methods are not sufficient when it comes to language comparison in typological research within general linguistics. Table 1 shows some of the crucial differences between the work with corpora in corpus linguistics and language comparison in language typology and historical linguistics. Note that we see this as tendencies, there are of course corpus linguists who work with spoken texts, for example. To highlight the differences, we would like to call the area of research which uses mathematical models on corpus data for language comparison and classification "Quantitative Language Comparison", as introduced by Michael Cysouw in his research group at the University of Munich[5]. The publications [15], [18] and [2] exemplify the kind of innovative approaches which are being developed in this emerging research field. Within this research area scientists work with annotated data from dictionaries and texts from a large group of different language families. They were mostly collected in language

[5] http://www.qlc.sprachwiss.uni-muenchen.de/index.html, accessed 27.8.2012

documentation projects or are the result of linguistic work in the field. The type of annotations range from extremely sparse annotations (only translations or chapters/verses in bible texts) to rich morpho-syntactic annotations manually added to audio and video transcriptions. The goal of the project Poio API is to make the data available to the existing and newly developed computational methods for analysis through a common and standardized representational mean, the annotation graph.

	Corpus Linguistics	Quantitative Language Comparison
Nr. of languages	1	>10
Orthography	standardized	different orthographies across sources
Mode	(mainly) written	spoken and written
Size of corpora	big (> 100.000 tokens)	small (around 10.000 tokens)
Annotations	more or less standardized (tagsets etc.)	different annotation schemes even within one project

Table 1: Tendencies Corpus Linguistics vs. Quantitative Language Comparison

5 Poio API

The framework we develop to accomplish the task of using a standardized representation is Poio API[6], a scientific software library written in Python. It provides access to language documentation data and a wide range of annotations types stored in different file formats. Poio API is based on a common and standardized representation format (LAF). The data and annotations can then be used with existing NLP tools and workflows and hence be combined with any other data source that is isomorphic to the representations in our framework.

5.1 Annotation Graphs, LAF and GrAF

Part of Poio API is an implementation of the ISO standard 24612 "Language resource management - Linguistic annotation framework (LAF)" ([14]). As representational file format we will use GrAF/XML (Graph Annotation Framework) as described in the standard. LAF uses the idea of annotation graphs ([3]) to represent linguistic data. Graphs can generally be seen as the underlying data model for linguistic annotations. [11] gives an overview and examples of how data from different sources may be mapped into a LAF representation through GrAF and how graphs can directly be used in analysis tasks on this combined data. GrAF is

[6]https://github.com/cidles/poio-api, accessed 27.8.2012

already the publication format for the Manually Annotated Sub-Corpus (MASC) of the Open American National Corpus ([13]). The American National Corpus also provides plugins for the General Architecture for Text Engineering (GATE[7]) and the Unstructured Information Management Architecture (UIMA[8]).Hence, data and annotation represented with GrAF may be used directly in well established scientific workflow systems ([12]). Another advantage of using GrAF for language documentation data is its radical stand-off approach, where data and annotation are completely separated from each other and may be collected and improved collaboratively in a distributed environment. Poio API will thus facilitate the integration of results from different teams and provide a way to work independently on a data set and with heterogenous annotation sources. Since the use of stand-off annotations is not yet common in language typology nor language documentation, we see Poio API as an innovative approach which will lead to new options in interdisciplinary collaborative linguistic research[9].

5.2 The CLARIN project

Poio API is developed as part of a project of the working group "Linguistic Fieldwork, Ethnology, Language Typology" of CLARIN-D, the German section of the large-scale pan-European "Common Language Resources and Technology Infrastructure" project (CLARIN[10]). The software library will be part of a web service and application which allow researchers to access, search, and analyze data stored in The Language Archive (at MPI in Nijmegen) together with local data or data from other sources which conform to the already developed CLARIN standard proposal "Weblicht" ([10]) or can be mapped onto LAF. Poio API itself is also the basis for two desktop software packages (Poio Editor and Poio Analyzer) which are already being used by researchers in language documentation projects to edit and analyze data and annotations. The three main blocks of the implementation of Poio API are:

- API Layer: provides unified access to language documentation data and uses the concepts that researchers understand instantly (i.e. do not hand out graphs, but interlinear text);
- Internal Representation: implements LAF, as described above;
- Parser/Writer Layer: handles the data from different file formats, input and output.

Specifically, Poio API will provide unified access to two of the most common data formats in language documentation: ELAN's EAF format and the file format

[7] http://gate.ac.uk/, accessed 27.8.2012

[8] http://uima.apache.org, accessed 27.8.2012

[9] For problems regarding approaches without radical stand-off annotations see for example [1], [5]

[10] www.clarin.eu, accessed 27.8.2012

of the software Toolbox. It will then supply the data in a representation consistent with the concepts of researchers in language documentations, for example representation of data and annotation in interlinear text. Figure 1 shows the architecture of the library and how it is embedded within the project. The big block with the label "Library" represents Poio API. It contains "LAF" (Linguistic Annotation Framework) as a generic representation in the center, for which we will use an implementation of GrAF. This representation is mapped on several file formats on the one side and on hierarchical data structures (see 5.3.1) on the other side. Both mappings will be implemented with a plugin mechanism, so that developers can easily attach new file formats or their own data structures.

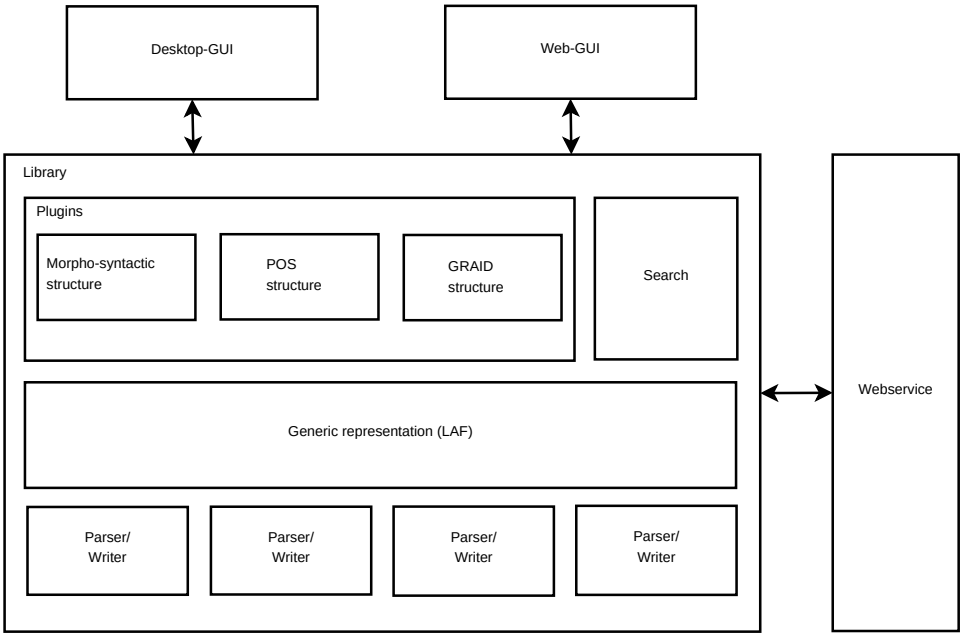


Figure 1: Architecture of Poio API

5.3 Technical implementation

Part of Poio API is based on the implementation of PyAnnotation[11], a library which allows researchers to access ELAN EAF and Toolbox files. This library has a similar goal as Poio API, but it does not use a general internal representation for the different annotation formats. This makes it difficult to add new types of annotation or other file formats. Poio API is a complete rewrite of PyAnnotation to extend the usage scenarios. We will first describe the two data types (data structure and annotation tree) the library currently supports to handle data and annotations and

[11] https://www.github.com/cidles/pyannotation, accessed 29.8.2012

then give an outlook on how we plan to connect the GrAF representation to those types.

5.3.1 Data Structure Types

We use a data type called *data structure type* to represent the schema of annotation in a tree. A simple data structure type describing that the researcher wants to tokenize a text into words before adding a word-for-word translation and a translation for the whole utterance looks like this:

```
[ 'utterance', ['word', 'wfw'], 'translation']
```

A slightly more complex annotation schema is GRAID (Grammatical Relations and Animacy in Discourse, [7]), developed by Geoffrey Haig and Stefan Schnell. GRAID adds the notion of clause units as an intermediate layer between utterance and word and three more annotation tiers on different levels:

```
[ 'utterance',
        [ 'clause unit',
        [ 'word', 'wfw', 'graid1' ],
      'graid2' ],
'translation', 'comment' ]
```

We see two advantages in representing annotation schemes through those simple trees. First, linguists instantly understand how such a tree works and can give a representation of "their" annotation schema. In language documentation and general linguistics researchers tend to create ad-hoc annotation schemes fitting their background and then normally start to create only those annotations related to their current research project. This is for example reflected in an annotation software like ELAN, where the user can freely create tiers with any names and arrange them in custom hierarchies. As we need to map those data into our internal representation, we try to ease the creation of custom annotation schemes that are easy to understand for users. For this we will allow users to create their own data structure types and derive the annotation schemes for GrAF files from those structures.

The second significant advantage is that we can directly transform the tree structures into a user interface for annotation editors and analysis software. Poio Editor and Analyzer make use of this and currently consist of no more than a few hundred lines of code but support every annotation scheme our data structure types can represent. This makes customization of the software for individual projects easier, as we remove a lot of complexity from our code base and can quickly introduce other software developers to our code.

We are aware that not all annotation schemes can be mapped onto a tree-like structure as in our data structure type. Non-linear annotations like co-reference or connections between tiers can not be represented with a simple hierarchical data type. We plan to support those schemes directly through the annotation graphs as represented in LAF and GrAF. We still have to find a simple strategy to map those annotation schemes to a graphical user interface later.

5.3.2 Annotation Trees

The data type *annotation tree* contains the actual content: data and annotations. The content is currently stored in a tree structure which mirrors the hierarchy of the data structure type. Figure 2 shows the relation between the data structure type and the annotation tree. Note that every open square bracket "[" in the data structure type has the implicit meaning "one or more elements of the following".

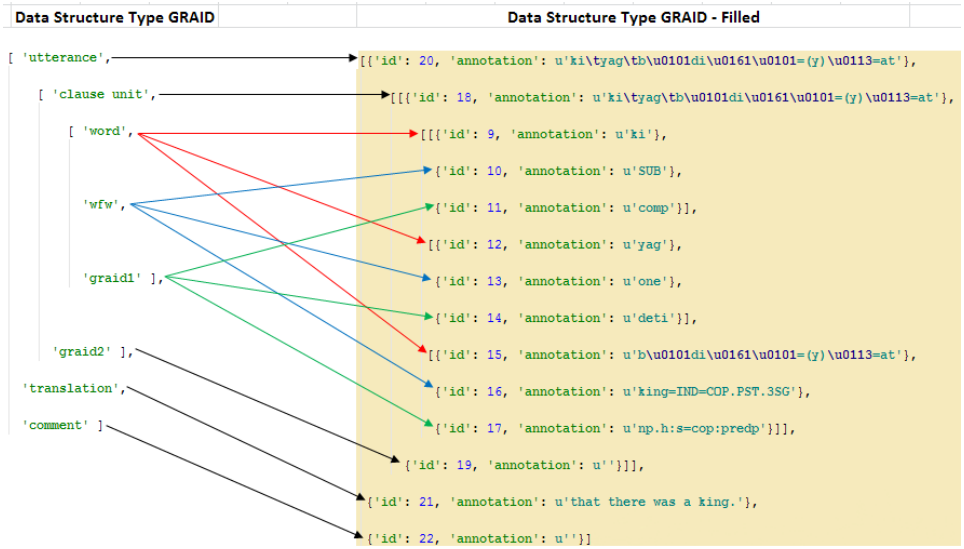


Figure 2: Relation between data structure type and annotation tree

The representation of the tiers containing tokenized data, such as the "clause unit" and "word" tier in the GRAID scheme, is still open to discussion. Right now they are given as full strings in the annotation tree, but we plan to return them as string ranges of the "utterance" tier. This reflects that tokens of the base data are stored by start and end indices in the annotation graphs. One problem is that those tokens might be represented by different strings as it is the case in the base data in some annotation schemes, for instance in a morpho-syntactic analysis. The following example shows how researchers encode implicit knowledge about morpho-phonemic processes in their annotations[12]:

(1) ref HOR068

tx	*Hegų*	*wogitekji*	***hųųroğoc***
mo	*hegų*	*woogitek-xjį*	***ho<į-Ø->roğoc***
gl	*that.way*	*be.angry-INTS*	***<1E.U-3SG.A>look.at***

[12]Example kindly provided by Prof. Dr. Helmbrecht from the University of Regensburg, selected from his Hocank [win] corpus.

wa'ų̨qkšąną, *hegų̨* *'eeja nų̨ų̨giwąkjį̨*
wa'ų̨-'ąk-šąną *hegų̨* *'eeja nų̨ų̨giwąk-jį̨*
do/be(SBJ.3SG)-POS.HOR-DECL that.way there run-INTS
kirikere *haa.*
kiri-kere *haa*
arrive.back.here-go.back.there make/CAUS\1E.A

ft He was looking at me real mad and I left there running fast.
dt 25/Sep/2006

Here the word *hų̨ų̨roǧoc* is still found as a token in the utterance tier (\t*x*), but the morphological analysis splits the word into the morphemes *ho-į̨-ǫ-roǧoc* which are not the same string as on the utterance tier. Those cases are easily stored in an annotation graph, as we can store the string representation of the morphemes in a feature vector of the token node or even attach a new node to it. We are currently working on an enriched version of the annotation tree which stores this additional information together with string ranges.

5.3.3 graf-python

The library *graf-python*[13] was developed by Stephen Matysik for the American National Corpus. It provides the underlying data structure for all data and annotations that Poio API can manage. The library graf-python is the Python implementation of GrAF. More information about GrAF, the corresponding Java implementation and how the framework implements annotation graphs can be found in the GrAF wiki[14].

GrAF comprises three important parts:

- A data model for annotations based on directed graphs;
- Serializations of the data model to an XML file;
- API methods for handling the data model.

The integration of GrAF in Poio API is still at an early stage, so we will not discuss it in detail here. The important question at the moment is how we can map the structure of an annotation graph into a data format which reflects the annotation schemes encoded by the data structure types. This intermediate data format will look similar to the annotation trees described above so that we can still feed the data to user interfaces and present the data to the researcher in a format he is familiar with.

Another open question is how we can transform the different file formats to a GrAF data structure. As mentioned above, the different tiers can be arranged in

[13] https://github.com/cidles/graf-python, accessed 30.8.2012
[14] http://www.americannationalcorpus.org/graf-wiki, accessed 30.8.2012

any way in a software like ELAN. We are currently working on different parsing strategies for those files to get the correct tokens and their annotations for the graph.

6 Conclusion

After 20 years collecting primary data on endangered languages and building multimedia and multi-purpose corpora, a new trend in Documentary Linguistics is emerging. The main focus lies now less on the documentation and more on the data, i.e. on the possible ways of combining and analyzing the collected data on a project-independent level. As we have shown in this paper, Poio API represents an important step in this direction.

References

[1] Bański, Piotr and Przepiórkowski (2009) Stand-off TEI annotation: the case of the national corpus of polish. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pp. 65–67.

[2] Bouda, Peter and Helmbrecht, Johannes (2012) From corpus to grammar: how DOBES corpora can be exploited for descriptive linguistics In *Language Documentation & Conservations Special Publication No. 4: Electronic Grammaticography*, Honolulu, Hawai'i: Department of Linguistics, UHM.

[3] Bird, Steven and Liberman, Mark (2001) A formal framework for linguistic annotation. In *Speech Communication*, Vol. 33, Issues 1–2, pp. 1–2, 23–60.

[4] Bird, Steven, Loper, Edward and Klein, Ewan (2009) *Natural Language Processing with Python.* O'Reilly Media Inc.

[5] Cayless, Hugh A. and Soroka, Adam (2010) On implementing string-range() for TEI. In *Proceedings of Balisage: The Markup Conference 2010* (URL: http://www.balisage.net/Proceedings/vol5/html/Cayless01/BalisageVol5-Cayless01.html, accessed 27.8.2012)

[6] Dwyer, Arienne (2006) Ethics and practicalities of cooperative fieldwork and analysis. In Gippert, Jost, Himmelmann, Nikolaus, Mosel, Ulrike (eds.) *Essentials of Language Documentation*, pp. 31-66. Berlin, New York: Mouton de Gruyter.

[7] Haig, Geoffrey and Schnell, Stefan Annotations using GRAID (2011) (URL: http://www.linguistik.uni-kiel.de/graid$_m$anual6.0$_0$8sept.pdf, accessed 30.8.2012)

[8] Himmelmann, Nikolaus (1998) Documentary and descriptive Linguistics. In *Linguistics 36*, pp. 161-195.

[9] Himmelmann, Nikolaus (2006) Language documentation: What is it and what is it good for?. In Gippert, Jost, Himmelmann, Nikolaus, Mosel, Ulrike (eds.) *Essentials of Language Documentation*, pp. 1-30. Berlin, New York: Mouton de Gruyter.

[10] Hinrichs, Marie, Zastrow, Thomas and Hinrichs, Erhard (2010) WebLicht: Web-based LRT Services in a Distributed eScience Infrastructure. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 19-21

[11] Ide, Nancy and Suderman, Keith (2007) GrAF: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, pp. 1–8, Prague, Czech Republic, June. Association for Computational Linguistics.

[12] Ide, Nancy and Suderman, Keith (2009) Bridging the gaps: interoperability for GrAF, GATE, and UIMA. In *Proceedings of the Third Linguistic Annotation Workshop*, pp. 27–34, Suntec, Singapore, August 6-7. Association for Computational Linguistics.

[13] Ide, Nancy, Baker, Collin, Fellbaum, Christiane, Fillmore, Charles, and Passonneau, Rebecca (2010) MASC: A Community Resource For and By the People. In *Proceedings of ACL 2010*, pp. 68–73, Uppsala, Sweden, July. Association for Computational Linguistics.

[14] ISO 24612:2012: Language resource management - Linguistic annotation framework (LAF) International Organization for Standardization, Geneva, Switzerland. (URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=37326 (accessed 27.8.2012))

[15] Mayer, Thomas and Cysouw, Michael (2012) Language comparison through sparse multilingual word alignment. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pp. 54–62, Avignon, France, April 23-24.

[16] McMahon, April and McMahon, Robert (2005) *Language Classification by Numbers.* Oxford: Oxford University Press.

[17] Russell, Stuart J. and Norvig, Peter (2003) *Artificial intelligence: A modern approach.* Upper Saddle River, N.J: Prentice Hall/Pearson Education.

[18] Steiner, Lydia, Stadler, Peter F., and Cysouw, Michael (2011) A Pipeline for Computational Historical Linguistics. In *Language Dynamics and Change*, pp. 89–127.

Syllabification of Middle Dutch

Gosse Bouma
Rijksuniversiteit Groningen
g.bouma@rug.nl

Ben Hermans
Meertens Institute
ben.hermans@meertens.knaw.nl

Abstract

The study of spelling variation can be seen as a window allowing us to understand the phonological systems of the dialects of Middle Dutch, and to what extent they differed. Syllabic information is of great help in the study of spelling variation, but manual annotation of large corpora is a labor-intensive task. We present a method for automatic syllabification of words in Middle Dutch texts. We adapt an existing method for hyphenating (Modern) Dutch words by modifying the definition of nucleus and onset, and by adding a number of rules for dealing with spelling variation. The method combines a rule-based finite-state component and data-driven error-correction rules. The hyphenation accuracy of the system is 98.4% and word accuracy is 97.4%. We apply the method to a Middle Dutch corpus and show that the resulting annotation allows us to study temporal and regional variation in phonology as reflected in spelling.

1 Introduction

The Corpus Van Reenen Mulder[1] (van Reenen and Mulder, 1993; Coussé, 2010) consists of Dutch legal texts from the 14th Century. It is one of the very few resources from that period that has been made available in electronic form. Due to its size and composition (over 2.700 documents dated between 1300 and 1399 and from all Dutch speaking regions in the Netherlands and Flanders) it is ideally suited for the study of spelling variation.

Our research is motivated by the need to study spelling variation of older Dutch texts. To a large extent spelling variation is determined by phonological systematicity; that is, differences in spelling can be a consequence of differences between dialects. Take for instance the difference between <*priester*>, <*preyster*> and <*prester*> *'priest'*. It is possible that the differences in the spelling of the vowel in the first syllable reflects a difference in the phonological status of this vowel in the respective dialects; a centralizing diphtong in the case of <*priester*>, a falling diphthong in the case of <*preyster*>, and a long mid vowel in the case of <*prester*>. This is a legitimate analysis, because in the Dutch dialect area as

[1] www.diachronie.nl/corpora/crm14

it is now, these differences are still attested. It is highly important to find cases where orthographic variation is determined by phonological systematicity, because they allow us to gain more insight into the phonological structure of the dialects of Middle Dutch. It would be the first step in the construction of a dialect atlas of the dialects of Middle Dutch.

To realize this goal the presence of information regarding syllable structure is of great help. With the presence of syllabic information it becomes a lot easier to determine the class of graphemes that are allowed in a specific position in a syllable, like the coda. That, again, might help us to gain more insight into the phonological processes that played a role in the domain of the coda. Consider, for instance, the fact that the grapheme <*gh*> is found fairly often in the coda position. Examples are <*meghtich*> *'mightly'* and <*ghetyghnesse*> *'testimony'*. Normally, the grapheme <*gh*> represents the voiced velar fricative. What does the presence of this grapheme in coda position tell us? It suggests perhaps that, in some Middle Dutch dialects, the process of devoicing did not apply in the same way it applies in the modern dialects. The study of the distribution of the graphemes representing voiced obstruents might reveal significant information regarding the processes of devoicing, voice assimilation and the difference between dialects of Middle Dutch with respect to these processes.

A second example of the importance of syllable structure is the distribution of <*l*> in the coda. An important difference between dialects is to what extent they allow the grapheme <*l*> in this position. In some dialects we find forms like <*arnolde*> (proper name) and <*goldene*> *'golden'*, whereas in other dialects we find the corresponding <*arnoude*> and <*goudene*>. The study of the distribution of <*l*> in coda position might reveal how exactly the sound change whereby */l/* vocalized to a glide developed over time, how it spread regionally, and in which phonological environments it took place.

In short, the study of spelling variation can be seen as a window allowing us to understand the phonological systems of the dialects of Middle Dutch, and to what extent they differed. By extenstion it allows us to study the development of phonological processes over time, and the way they spread regionally. The information of syllabic information is of great help in the study of spelling variation, and this is the reason why we think it is important to construct a method for automatic syllabification of words in Middle Dutch texts.

In this paper, we present a method to automatically add syllable boundaries to Middle Dutch words. The method adapts an existing method for hyphenating Modern Dutch words to Middle Dutch and consists of two parts: a finite-state transducer which implements the two main rules for Dutch hyphenation, and a statistical component that automatically learns rules to correct errors in the output of the first method.

For testing and evaluation as well as for automatically learning error-correction rules, we created a gold standard list of hyphenated word types. To this end, 50% of the word types from the corpus was hyphenated automatically using the rule-based finite-state system. The output of the automatic system was manually corrected

to obtain a gold standard. Note that given the limited size of the corpus (approximately 650.000 tokens and 43.000 types),[2] we could also have corrected all types and used the corrected list to hyphenate running text. The advantage of our automatic method over a dictionary-based method, however, is that it will also be able to hyphenate words that are not in the dictionary, and thus, its coverage on unseen corpus-data will probably be better than a method based only on dictionary look-up. In practice, highest accuracy is probably obtained by applying dictionary look-up for known words, and the automatic method for unknown words.

In the context of two recent projects (Adelheid[3], InPolder[4]) the texts in the corpus van Reenen Mulder have been annotated with lemmas (in Modern Dutch spelling) and morphological structure. Syllable boundaries, however, are not indicated. As there is a considerable gap between the original spelling and the corresponding lemma's in Modern Dutch, it is not easy to determine syllable boundaries or hyphenation points on the basis of a hyphenated Modern Dutch dictionary.

Below, we describe previous work on hyphenating (Modern) Dutch and relevant differences between Modern and Middle Dutch. Next, we present our implementation of the finite-state hyphenation method, which achieves a hyphenation accuracy of 94.0%. In section 5, we apply transformation-based learning and improve accuracy to 98.4%. In section 6, we give some examples of using the corpus for studying temporal and regional tendencies in spelling variation.

2 Hyphenating Modern Dutch

Bouma (2003) describes a method for accurate hyphenation of Modern Dutch text. It consists of two steps: a finite-state transducer that implements the maximum onset principle, the most important rule for hyphenating Dutch words, and a transformation-based learning component that, given a word list of correctly hyphenated words, automatically learns rules to correct errors produced by the finite-state transducer. The system is trained and evaluated on hyphenated word forms obtained from CELEX. The reported hyphenation accuracy (i.e. percentage of correctly inserted hyphens) is 99.3% and the word accuracy (i.e. percentage of correctly hyphenated words) is 98.2%.

Below, we give an informal overview of the system, emphasizing those aspects that will need reconsideration for our current task.

The rules for syllabifying words in Dutch follow two general principles:

1. Syllable boundaries cannot cross morpheme boundaries.

2. The maximum onset prinpiciple is respected. That is, consonants that may be added to either a preceding or following syllable are added to the onset of

[2] This is an approximation, as the transcription contains diacritic tokens to indicate unclear parts of the original manuscripts, words written as one, etc. We ignored those in our counts.

[3] http://adelheid.ruhosting.nl

[4] http://depot.knaw.nl/8914/

the following syllable if this does not violate constraints on onset clusters.

The finite-state method for hyphenating words only implements the second constraint. The reason for ignoring the first constraint is that detecting morpheme boundaries is hard, and requires, at least, a detailed lexicon and an implementation of morphological rules. The second constraint can be implemented using finite-state techniques. A detailed description is in Bouma (2003). That solution involves the following steps:

1. Mark the beginning of a word (represented as a sequence of characters).
2. Mark the beginning and end of each nucleus in a word.
3. Insert a hyphen at each position between a nucleus and a following nucleus, in such a way that the onset of the second nucleus is maximal.
4. Remove all markers except the hyphens.

Each of these steps can be implemented as a finite-state transducer, and the resulting system is then the composition of these transducers. The implementation of the rules is greatly simplified by the *replace* operator (Karttunen, 1995; Gerdemann and van Noord, 1999), a finite-state method for implementing phonological rules (i.e. contextually sensitive rules for replacing one symbol sequence by another).

In step 2, the method requires that each *nucleus* is marked. A nucleus is defined here as the maximal sequence of characters (going from left to right) that can represent a vowel or diphtong in Dutch. To this end, a listing of all possible nucleus character sequences is provided. Similarly, in step 3, the method requires that the onset of the second syllable follows the rules of Dutch orthography. Again, this is implemented by providing a list of all possible onset character strings. No constraints are imposed on the coda of a syllable, other than that it must consist of a sequence of consonants. An example of this algorithm for the word <*aardappel*> *'potato'*, a compound of <*aard*> *'earth'* and <*appel*> *'apple'* is shown below:

```
aardappel
    ⇓
+aardappel
    ⇓
+@aa@rd@a@pp@e@l
    ⇓
+@aa@r-d@a@p-p@e@l
    ⇓
aar-dap-pel
```

Note that the sequence <*aa*> is marked as a single nucleus, in spite of the fact that a letter <*a*> can also form a nucleus by itself. The output of the system contains an error, as it identifies `aar-dap` as a syllable boundary, where this should

have been `aard-ap`, following the morphological boundary between <*aard*> and <*appel*>.

The hyphenation accuracy of a system that only uses the notions nucleus and onset, is 94.5%, and word accuracy is 86.1%. To improve accuracy, one can use data-driven machine learning techniques that learn from correctly segmented words. In Bouma (2003), a method is presented that uses transformation-based learning (Brill, 1995; Ngai and Florian, 2001). Given a word, the system considers both the hyphenation predicted by the finite-state system and the correct hyphenation produced by a human expert. By inspecting large data samples, the system learns rules that correct frequent errors in the system output. In Bouma (2003), 290.000 hyphenated words from CELEX are used for training, and a hyphenation accuracy of 99.3% and a word accuracy of 98.2% is achieved. This accuracy is comparable to that of state-of-the-art hyphenation methods, such as the hyphenation patterns implemented in the text typesetting package LaTeX. An interesting feature of transformation-based learning is that the error-correcting rules can themselves be interpreted as finite-state transducers, and thus can be composed with the baseline finite-state hyphenator to obtain a highly efficient and accurate finite-state hyphenator.

3 Challenges

Adapting the method described above to Middle Dutch requires that we modify the definition of nucleus and onset and deal with some peculiarities of Middle Dutch spelling.

Middle Dutch texts exhibit a substantial amount of spelling variation, as illustrated in table 1. As a consequence, nuclei and onsets will also exhibit a wider range of variation than in Modern Dutch. The nucleus <*ey*> in <*borghermeyster*>, for instance, does not exist in Modern Dutch. More in general, the spelling of long vowels sometimes involves doubling of the character (as in Modern Dutch), but in other cases addition of <*e*>, <*i*> or <*y*> Thus, we find <*aan*>, <*aen*>, <*ain*>, and <*ayn*>.[5]

While spelling variation in itself does not make the hyphenation task harder (it just requires adding alternative spellings of nuclei and onsets), there are some patterns that do require special attention:[6]

- The characters <*i*> and <*j*> are interchanged frequently (<*iaer*> vs <*jaer*> *'year'*, <*ighelic*> vs. <*jghelic*> *'in fact'*).
- Similarly, <*u*> and <*v*> are often used interchangeably: <*uerclaringhen*>

[5] Van Halteren et al., `www.ccl.kuleuven.be/CLARIN/vanhalteren.pdf`, and Kestemont et al. (2010) address the issue of spelling normalization, where tokens in the original text are linked to the most likely lemma in Modern Dutch.

[6] These are all a consequence of the fact that Middle Dutch orthography is influenced by Latin, which did not distinguish between <*i*> and <*j*>, <*v*> and <*u*>, and did not have a <*w*>.

borgchermestere	burgeremeystere	burghermeisters
borgermestere	burgermeesteren	burghermestere
borghemeestere	burgermeister	burghermeysters
borghemeyster	burgermeystere	burghmeester
borghermeester	burghemeesteren	burghmeisters
borghermeistere	burghemeisteren	burghmesters
borghermester	burghemeysteren	burghmeysteren
borghermeyster	burghemeysters	burghmeysters
borhermestere	burghermeestere	burgmesters
burchmeester	burghermeistere	

Table 1: Spelling variants of the Dutch word for *'mayor'*. Inflected forms are only included if the uninflected form with the same spelling was absent.

vs. <*verclaringhen*> *'statement'*, <*uerstaen*> vs. <*verstaen*> *'understand'*, <*zeuentien*> vs. <*zeventien*> *'seventeen'*.

- The letter <*w*> is often used for the diphtong <*uu*> (<*uutghesproken*> vs. <*wtghesproken*> *'stated'*, <*zuutzide*> vs. <*zwtzide*> *'southside'*).
- Finally, double <*v*> is sometimes used to denote a vowel (long <*u*>) (as in <*hvvs*> (*'house'*)) or a consonant (<*w*>) as in <*gesvvoren*> (*'sworn'*).

This makes syllabification hard, as <*i*>, which only is (part of) a nucleus in modern spelling, may also be (part of) an onset in Middle Dutch. Similarly, <*j*> is only used in onsets in Modern Dutch (except for the diphtong <*ij*>), but can also be (part of) a nucleus in Middle Dutch. The same ambiguity holds for the character pairs <*u*> and <*v*>, the character <*w*> and the character bigram <*vv*>.

4 Adapting a Rule-based Hyphenator

As the general principles of syllabification have not changed, the general architecture of the rule-based, finite-state, hyphenator for Modern Dutch outlined in section 2 can remain unchanged. To account for the difference in orthography, however, we must adapt the definition of nucleus and onset. Furthermore, characters `i`, `j`, `u`, `v`, and `w` require special attention.

4.1 Spelling issues

To make the task of identifying nuclei and onsets more accurate, we first replace the character `u` with `U` in those cases where `u` functions as consonant, and replace `v`, `w`, and `j` with `V`, `W` and `J` respectively, in contexts where these function as vowels:

nucleus a, aaC, ae, ai, au, e, ee, ei, eu, ey, i, ie, ii, iae, J, o, oe, ooC, ou, oi, oy, u, uuC, ue, uy, ui, V, Vy, W, y, ye

onset b, bl, br, c, ch, cl, cr, d, dr, dw, f, fl, fr, g, gh, gl, gr, h, j, k, kl, kn, kr, l, m, n, p, ph, pl, pr, Q, r, s, sc, sch, schr, scr, sl, sn, sp, spl, spr, st, str, t, th, tj, tr, U, v, vl, vr, w, wr, x, z, zw

Table 2: A listing of the definition of nucleus and onset. Upper case C denotes any consonant. Upper case Q denotes the letter combination qu. Upper case J,U,V,W indicate occurrences of lower case j,u,v,w that function as vowel (<*j*>,<*v*>,<*w*> or consonant<*u*>).

- In the sequences aue, eue, and oui, u almost always functions as a v. Therefore, we replace such sequences with aUe, eUe, and oUi, respectively, where we use U as the character that denotes a u functioning as a consonant.
- In the sequences C_1vC_2 and $+vC_2$, v almost always functions as u. (We impose some restrictions on C_2 to prevent over-generalization.) We replace the v in such sequences with V to denote a v functioning as a vowel character.
- In the sequences C_1wC_2 and $+wC_2$, w almost always functions as uu. We replace the w in such sequences with W to denote a w functioning as a vowel character.
- In the sequences $+jn$ and $+jm$, j functions as i. Therefore, we replace j with J to denote a j functioning as vowel.

After these rules have been applied, the steps that identify nuclei and syllable boundaries are applied as in the finite-state method presented above.

4.2 Updating definitions

A nucleus is represented orthographically as a sequence of one, two, or sometimes three vowel characters. Middle Dutch spelling allows for some sequences that are not used in Modern Dutch (i.e. <*ae*>, <*ai*>, <*ey*>, <*ii*>, <*iae*>, <*oy*>, <*uy*>, <*ye*>). We inspected the word list to find the most frequent cases and incorporated these in the definition of nucleus, as shown in table 2. The spelling of onsets is to a large extent identical to that in Modern Dutch.

The nuclei aaC, ooC and uuC in the list are not quite in accordance with linguistic notions, but were introduced to make the prediction of hyphenation points more accurate. In Modern Dutch, long vowels in closed syllables are represented by a double character <*aa*>, <*oo*> or <*uu*>.[7] Therefore, if we encounter such

[7]The long vowel <*ee*> is an exception, as this can also occur in word-final open syllables, i.e. <*zee*> *'sea'*.

a sequence, we know the syllable of which it is the nucleus has to contain a non-empty coda. By marking the following consonant as part of the nucleus in such cases, we prevent the maximum onset principle from considering the consonant as part of the onset of a following syllable. The problem can be illustrated with a word like <*clooster*>, *'monastry'*. By recognizing a nucleus `oos`, we predict the hyphenation `cloos-ter`. Had we predicted the nucleus `oo`, we would obtain the hyphenation `cloo-ster`, as <*st*> is a possible onset. It should be noted, though, that the rule that says that <*aa*>, <*oo*>, and <*uu*> are always followed by a non-empty coda is not absolute (in contrast with Modern Dutch). So, we do also find examples such as <*coo-pen*> and <*boo-de*> and thus this strategy is less accurate in Middle Dutch than in Modern Dutch.

4.3 Implementation

We implemented the hyphenation system as a sequence of finite-state transducers. The steps in the algorithm are the same as for Modern Dutch, except that we introduce one additional step where certain letters are replaced by upper case letters in order to make the following steps more accurate.

1. Mark the beginning of a word (represented as a sequence of characters).
2. Replace letters `j,u,v,w` and bigram `qu` with an upper case letter in certain contexts.
3. Mark the beginning and end of each nucleus in a word.
4. Insert a hyphen at each position between a nucleus and a following nucleus, in such a way that the onset of the second nucleus is maximal.
5. Remove all markers except hyphens, and convert upper case letters to their lower case counterparts.

The most crucial parts are steps 3 and 4, which require a definition of nucleus and onset. We used the definitions in figure 2. We arrived at these definitions by iterative testing on the list of types from the corpus.

The manually developed rule-based system is useful mostly as a base-line system that helped us in creating a manually corrected list of word types with hyphenation points. That is, we compiled a word list from our corpus and automatically hyphenated the word types using our finite-state hyphenator. 50% of the resulting data was corrected by a human expert. As the accuracy of the rule-based system is relatively high, the amount of manual labor was modest and far more efficient than an approach where hyphenation patterns have to be added manually to all words.

After creation of the gold standard word list, we observed that the automatic system in fact achieved 94.0% hyphenation accuracy and 90.1% word accuracy.

word	(*answered*)	a n d w e r d e
system	*an-dwer-de*	0 0 1 0 0 0 1 0
correct	*and-wer-de*	0 0 2 0 0 0 1 0
word	(*pilgrimage*)	b e d e v a e r d
system	*be-deu-aerd*	0 0 1 0 0 1 0 0 0
correct	*be-de-uaerd*	0 0 1 0 0 9 0 0 0

Table 3: Aligning system output and correct hyphenation patterns

5 Applying Transformation-based learning

By comparing the output of the automatic finite state hyphenation program with the correct hyphenation in our segmented word list, we can detect where the program makes errors. If we can find patterns or regularities in the errors, we can try to add rules to the hyphenator that would correct or prevent these errors.

Transformation-based learning Brill (1995); Ngai and Florian (2001) is a machine learning method that automatically tries to find the rule that corrects most errors in the data. The score of a rule (on a set of training data) is the number of correctly corrected errors minus the number of newly introduced errors (as the rule will usually also apply to a number of cases that were actually correct). The method has been used mostly in part-of-speech tagging, and requires annotated data for training. Given a corpus that is both manually annotated (providing the gold standard) and annotated by a baseline POS tagger (for instance, a system that always assigns the most frequent POS for a word), TBL learns rules that replace an incorrect POS-tag in the system output by a correct POS-tag. After applying the rule with the highest score to the output of the baseline system, rule scores are recomputed and the rule with the highest score on the modified data is applied, and so on until a stopping criterion is reached.

Correcting hyphenation errors can be seen as a similar task. We first assign to each letter in an input word either the value 0 or 1, where 0 stands for 'not preceded by a hyphen' and 1 stands for 'preceded by a hyphen'. Furthermore, the gold-standard hyphenation pattern is aligned with the system output by using two more codes, 2 in case the hyphen actually has to be placed one more position to the right, and 9 in case a hyphen has to be placed one position to the left. As explained in Bouma (2003) this coding has the advantage that the correction of a hyphenation mistake in general requires 1 instead of 2 correction rules. Two examples of this alignment scheme are given in table 3.

We performed 10-fold cross-validation, where in each experiment the system is trained on 90% of the data, and test on the remaining 10%. The highest scoring rules for one of the experiments are given in table 4.

After applying the rules to a held out portion of the data, we obtain an average hyphenation accuracy of 97.90% (s.d. = 0.173) and an average word accuracy of

-ster	→	s-ter	eu-a	→	e-ua
-ru	→	r-u	+io-	→	+io
-fl	→	f-l	c+hei	→	ch-ei
-y	→	y	l-ue	→	lu-e

Table 4: The highest scoring error correcting rules learned by TBL

96.5% (s.d. = 0.317).

Roche and Schabes (1997) have shown that the rules learned by TBL can be interpreted as finite-state transducers that replace a symbol by another symbol in a given context. Sequential rule application corresponds to the composition of the transducers for the individual rules. An end-to-end system, finally, that hyphenates words, can be obtained by the composition of the finite-state transducer that represents the rule-based system and the finite-state transducer that implements all TBL rules. Using the FSA tools, we have computed this finite-state transducer (131 states, executable is 442 kB.)

6 Preliminary Exploration of results

The segmentation program that is the result of combining the rule-based and error-driven components can be applied to all of the texts in the Corpus van Reenen Mulder (CRM14). The result is texts in which each word has been segmented into syllables with an accuracy of over 97%.[8]

The metadata in CRM14 provides both the year and location of the manuscripts. Thus, we can easily compute the relative frequency of syllables, onsets, or nuclei over a period of 100 years (1300-1400). For instance, onsets <*gh*> and <*g*> are often used interchangeably, as witnessed by the fact that we find over 3000 minimal pairs, i.e. word types that differ only in the choice for <*gh*> vs. <*g*> such as <*al-ghe-heel*> vs. <*al-ge-heel*>. Similarly, the nuclei <*ei*> and <*ey*> are often used interchangably (over 700 minimal pairs).

Figure 1 compares the change in relative frequency of the onsets <*gh*> and <*g*> over time. It shows that during the period represented in the corpus, the frequency of onset <*gh*> is falling while the frequency of onset <*g*> rises. Note that for this comparison, obtaining accurate frequency estimates from a corpus with unsegmented words is challenging, as determining whether *g* is part of a coda or onset in word-internal positions is hard without actually segmenting the word.

We can also study regional tendencies in spelling. The geographical distribution of <*ei*> as a percentage of all <*ei*> + <*ey*> occurrences for a given location,[9]

[8]The word accuracy on running text is probably higher than that for a word list for the same text. One reason is the fact that monosyllabic words tend to be frequent. As the programme hardly makes mistakes on monosyllabic words (as identification of a nucleus is close to perfect), the overall accuracy on running text will go up.

[9]Only shown for locations for which at least 10 occurrences of both forms were available.

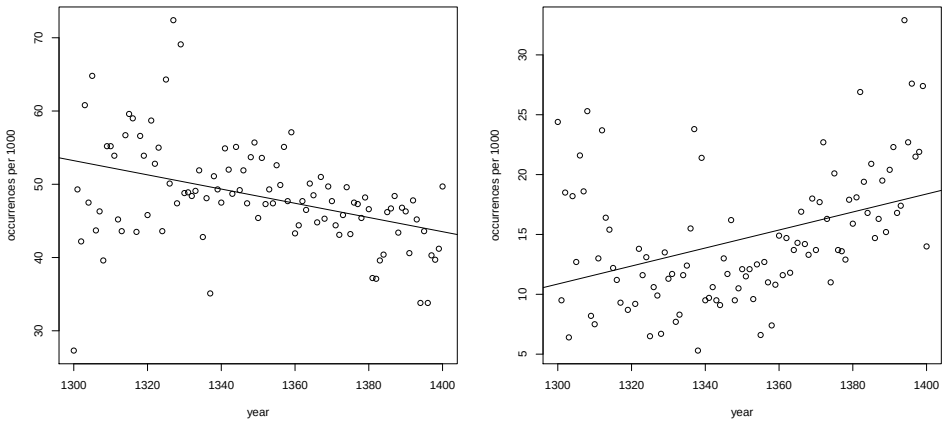
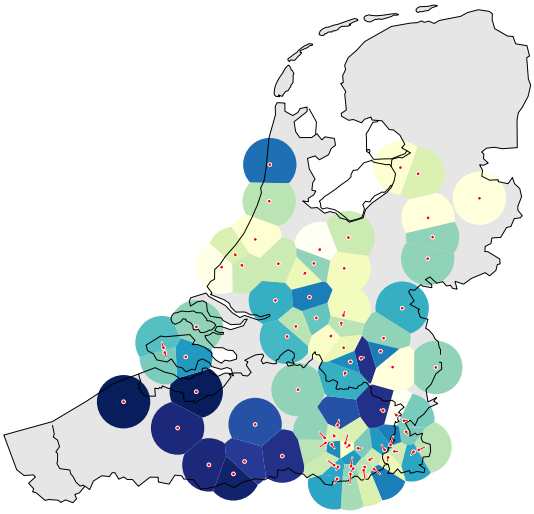


Figure 1: Distribution of the onsets *gh* (left) and *g* (right) over time (1300-1400).

Figure 2: <*ei*> occurrences as a percentage of the total number of <*ei*> and <*ey*> occurrences for various locations. Darker colors indicate a higher percentage of <*ei*>.



is given in Figure 2. It shows that preference for <*ei*> was stronger in the south of the Netherlands than in the north.[10]

7 Conclusion

The study of phonological processes in historical text can benefit from accurate information about the morphological and phonological structure of words. We have presented a method for accurate syllabification of Middle Dutch texts, using a finite-state and data-driven method originally developed for Modern Dutch. The result can be compiled into an efficient transducer that can be used to automatically

[10]The map was created using the Gapmap software, `www.gabmap.nl`.

annotate large corpora from the given era with syllable boundaries. An obvious candidate is the Corpus Gysseling.[11]

By applying this method to the complete CRM14, we obtain a corpus annotated with syllable boundaries. We demonstrate that this information can be used to study both temporal and regional variation in the distribution of onsets and nuclei. In future work, we hope to show that this can be the basis of deeper and more principled studies into the phonology of Middle Dutch.

References

Gosse Bouma. Finite state methods for hyphenation. *Journal of Natural Language Engineering*, 9:5–20, 2003. Special Issue on Finite State Methods in NLP.

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21: 543–566, 1995.

Evie Coussé. Een digitaal compilatiecorpus historisch Nederlands. *Lexikos*, 20: 123–142, 2010.

Dale Gerdemann and Gertjan van Noord. Transducers from rewrite rules with backreferences. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 126–133, Bergen, 1999.

Lauri Karttunen. The replace operator. In *33th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Boston, Massachusetts, 1995.

M. Kestemont, W. Daelemans, and G. De Pauw. Weigh your words-memory-based lemmatization for Middle Dutch. *Literary and Linguistic Computing*, 25(3): 287–301, 2010.

Grace Ngai and Radu Florian. Transformation-based learning in the fast lane. In *Proceedings of the second conference of the North American chapter of the ACL*, pages 40–47, Pittsburgh, 2001.

Emmanuel Roche and Yves Schabes. Deterministic part-of-speech tagging with finite-state transducers. In Emmanuel Roche and Yves Schabes, editors, *Finite state language processing*, pages 205–239. MIT Press, Cambridge, Mass., 1997.

P. T. van Reenen and M. Mulder. Een gegevensbank van 14de-eeuwse Middelnederlandse dialecten op de computer. *Lexikos*, 3:259–281, 1993.

[11] `http://gysseling.corpus.taalbanknederlands.inl.nl/cqlwebapp/search.html`

Casting a Spell: Identification and Ranking of Actors in Folktales

Folgert Karsdorp, Peter van Kranenburg, Theo Meder,
Antal van den Bosch
Email: {folgert.karsdorp,peter.van.kranenburg,
theo.meder}@meertens.knaw.nl
a.vandenbosch@let.ru.nl

Abstract

We present a system to extract ranked lists of actors from fairytales ordered by importance. This task requires more than a straightforward application of generic methods such as Named Entity Recognition. We show that by focusing on two specific linguistic constructions that reflect the intentionality of a subject, direct and indirect speech, we obtain a high-precision method to extract the cast of a story. The system we propose contains a new method based on the dispersion of terms to rank the different cast members on a scale of importance to the story.

1 Introducing the Problem

Today do I bake, tomorrow I brew,
The day after that the queen's child comes in;
And oh! I am glad that nobody knew
That the name I am called is Rumpelstiltskin!

So says the song of the little man in Brother Grimm's *Rumpelstiltskin* with which he accidentally reveals his name to the queen. Advances in the task of Named Entity Recognition (NER) make it possible for computer systems to recognize his name as well. NER seeks to locate and extract atomic textual units into a predefined set of categories (e.g. PERSON, LOCATION and ORGANIZATION).

Now that we can recognize *Rumpelstiltskin* as being the name of a person, how do we know that he is a character or actor in the story? Are all named entities within the category of PERSON in a story automatically actors? How about characters that have no proper name in a story, but are referred to by nominals or noun phrases? How do we know that they are part of the cast of a story?

Most NER systems are developed for the domain of non-fiction, including newspapers, manuals and so forth. Applications of NER in literary texts or folktales, however, have been discussed only marginally (see [13] for an exception).

Unfortunately, the knowledge gained from traditional NER systems is not easily transferred to the domain of fictional texts, because of several incongruences between both domains. Named entities in fairytales, for example, are represented in ways quite different from those in non-fiction. We find traditional proper nouns such as *Hansel* and *Gretel* but also noun phrases such as *the king* and *the big bad wolf* which, although expressed differently, have the same function as their proper noun counterparts. To make things even more complicated, inanimate entities in folktales may belong to the cast as well. A nice example of this can be found in the story of *The Fleeing Pancake*, in which a pancake acts as the protagonist who tries to escape from its predators.[1]

In this paper we present a method for extracting the cast from fictional texts. The cast consists of a number of actors which we, following Bal [1], define as "agents that perform actions" and have intentions. An action is seen as to cause or experience an event. Actors are thus not necessarily human. As claimed by structuralists such as Greimas, actors are distinguished from other entities by having an intention. They strive towards a goal or aspire a particular aim [1]. The identification of actors in a story therefore would require a method for detecting intentionality. We show that direct and indirect speech are effective indicators of intentionality. Extracting these constructions from a text gives us a high precision method for retrieving the actors that form the cast of a story.

Not all actors in a story are equally important. The scale of importance ranges from what is customary referred to as 'the hero' of a story to mere background actors whose actions have relatively little influence on the course of the story. The system we propose ranks the different cast members on a scale of importance to the story on the basis of their dispersion in the text. We show that the ranking of the system corresponds to a large extent with that of expert knowledge.

The outline of the paper is as follows. We begin with some background information about actors from the perspective of narratology and combine this with insights from linguistics in Section 2. We then describe the corpus used and the way the annotations of the corpus have been established. Section 4 presents the computational methods used in this study. In Section 5 we report on the empirical results for the proposed method. The last section offers our conclusions and directions for further research.

2 Backgrounds

2.1 Narratological Background

We ground our definition of an actor in the theory of narratology as developed by Bal [1] who borrows from the French structuralist Greimas [6]. The underlying presupposition in the proposal by Greimas is that human activity (thinking and action) targets some aim. Actors are no different and show some intentionality of

[1] See for example `http://www.pitt.edu/~dash/type2025.html#gander`

achieving this aim or goal. This could be the achievement of something pleasant, some object of desire (be it physical or non-physical) or something disagreeable. Natural language reveals this intentionality in particular verbs (e.g. *to fear*, *to wish*).

Bal distinguishes six classes of actors. There are actors who pursue some aim (called SUBJECT) and the aim itself (called OBJECT). A SUBJECT *x* strives towards some aim *y* where *y* is an OBJECT. In a prototypical fairytale love story these roles might be filled with 'the prince who wants to marry the princess'. OBJECTS are not necessarily human or even animate but could also be a mental state or an aspiration such as 'becoming king'. Other types of actors in Bal's theory are POWER and RECEIVER. POWER is often an abstraction that supports the SUBJECT in reaching its goal or prevents it from getting there. The RECEIVER is the one to whom some desired object is 'given'. These two actor types are closely related to another pair of actors, namely the HELPER and the OPPONENT. As their name reveals, HELPERS help the SUBJECT in achieving its goal (although providing only incidental help) when some OPPONENT intends to distract it from doing so.

Actors are thus not necessarily human or animate beings, but can be of a more abstract nature as well, such as skills and abilities. In this paper we are only interested in the classes of actors that show intentions in the context of a particular story. Whether they are animate or not is only partly based on reality and is determined mainly in relation to the story itself.

2.2 Linguistic Background

A baseline approach to the problem would be to tag all animate nouns as actors. However, this compromises both on precision and recall, because not all animate entities are actors in a story and not all inanimate entities are not actors.[2] Following [1], we claim it is better to look for entities that exhibit intentionality or consciousness. Are there any linguistic clues for intentionality?

Sentences in which the intentionality of some entity is most clearly revealed are sentences containing direct speech. In traditional grammar the quoted clauses in the examples below are commonly characterized as (direct) objects of the verbs:

(1) *"Lay your head on the chopping block," said the old witch.*

(2) *"I heard the world cracking and I flee away," answered the goat.*

In sentences like these we read the text as if it was uttered by the actor. That we should do so is both linguistically (by means of syntax) and orthographically (by means of quotes) marked. However, speech can also be indirectly narrated, just as thoughts, as the following examples show:

(3) *The man asked why the old woman smashed all her eggs to pieces.*

[2]Moreover, the identification of animate nouns would require a semantic lexicon, which is a rather expensive resource not available for all languages.

(4) *The robbers thought the farmer had nothing to hide.*

In these examples of indirect speech it is less clear that the presented text has to be attributed to the actor; it may by influenced by the narrator's stance towards the situation. Traditionally, these sentences are characterized as complement constructions in which the subordinate clauses fill the role of (direct) objects of the verbs of the main clause. Most verbs that take such 'direct object clauses' belong to one of the following semantic types [7]:

i Verbs that express a command, question, promise, etc (i.e. verbs of communication such as *to tell, to promise, to beg*);

ii Verbs that denote knowing, believing, supposing, etc (*to imagine, to understand, to realize*);

iii Verbs that indicate the evaluation of something (*to appreciate, to regret*);

iv Verbs that express a wish or desire (*to wish, to hope*);

v Verbs that denote a particular way of perception (*to hear, to discover, to see*).

Verhagen points at an interesting commonality between these five verb types:

> "[...] such predicates all evoke a mental state or process of a subject of consciousness (sometimes a process *comprising* a mental state, as in the case of communication), and the content of the complement is associated with this subject's consciousness in a particular manner." [15, 100]

The resemblance with the idea of Greimas that actors are intentional entities aspiring a particular goal or aim, should be clear. If we can automatically retrieve the subject of the main clause of a complement construction, we have a good chance of finding an actor in a story. Both instances of direct speech and indirect speech provide us with cues about the intentionality of entities without the need to know the verb's semantics.

2.3 Computational Background

Few studies have focused on the recognition of actors in fictional texts. Declerck *et al.* [2] present an ontology-based method for detecting and recognizing characters in folktales. They concentrate on the interaction between a hand-crafted ontology of folktale characters and the linguistic analysis of indefinite and definite nominal phrases. Although promising, given the rather small scale of the experiment reported in this study, care must be taken when interpreting and generalizing the results. Using both syntactic features and a semantic lexicon, Goh *et al.* [5] propose an automated approach to the identification of the main protagonist in fictional texts, in particular fairytales. The method is not extended to the rest of actors.

Several studies have tried to tackle the problem of quote attribution. Quote attribution is the task of automatically extracting quoted speech and assigning them to their speakers. This task is interesting to us since we assume that the speakers of quoted speech and the conceptualizers of indirect speech form the cast of a story.

The studies about quote or speaker attribution can be divided into two types of approaches: rule-based approaches and machine-learning approaches. Rule-based approaches are represented by e.g. [4, 11, 3], who on the basis of syntactic rules and finite-state automata attempt to select the correct speaker of a piece of quoted speech. Other studies make use of machine-learning methods, e.g. [10, 3]. To find the correct speaker of a quote, Elson and McKeown [3] assign the quotes into one of seven syntactic categories. Many of these categories unambiguously lead to the identification of the speaker. For the remaining categories, a binary classifier is used that predicts for each pair of candidate speaker and quote whether it is the speaker or not on the basis of a feature vector. In contrast with most other work, Ruppenhofer *et al.* [11] incorporate not only direct speech, but also indirect speech and is thus more related to our work.

Most of the approaches mentioned above make use of external semantic lexicons to distinguish nominal phrases that are potential characters from other nominal phrases. The primary selection criterion is whether the nominal phrase falls within the category of living things. However, as stated above, whether some entity is animate or not is determined in relation to the story itself. The same lexicons are used to select sentences containing verbs of communication. In our approach we can do away with such external sources by formulating specific linguistic constructions. In doing so, our method is not restricted to, for example, verbs of communication, and can identify the more general category of verbs that describe some other mental state such as a thought, a belief or a desire. Another difference is that our approach is not limited to speaker identification of directly quoted speech, but incorporates the conceptualizer of indirect speech as well. Finally, our system is not restricted to the identification of an externally defined set of animate actors, but can in principle identify any subject of consciousness.

The method presented in this study not only tries to identify actors, but also attempts to rank them on a scale of importance. To our knowledge there are no computational studies that model the importance of actors in fictional texts.

3 Corpus, its annotation and tools

Corpus The corpus consists of a collection of 78 Dutch folktales from [12]. The collection is part of the Dutch Folktale Database from the Meertens Institute.[3] The collection was selected because it represents a homogeneous set of texts: (1) all texts are fairytales, (2) they are written in standard Dutch and (3) they are edited by the same author. The average number of words per story is 824 and a story consists of 44 sentences on average.

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

Annotation To obtain gold-standard annotations of which actors are present in the texts, we asked two experts[4] in folktale research from the Meertens Institute to give a list of the actors of the stories. In addition to this list they were asked to rank the actors on a scale of importance to the story. Actors that were thought to be of equal importance should be placed in the same position in the ranked list. If a particular actor in a story is referred to with multiple names, we asked the experts to list all alternative names together in single entries in the ranking. We did not attempt to use or develop automatic procedures to resolve these coreferences.

Preprocessing and tools All texts were processed using the Dutch morpho-syntactic analyzer and dependency parser *Frog* developed at the ILK Research Group [14].[5] Frog contains a tokenizer that offers a high-precision quote detection system which was crucial for extracting all quoted sentences from the stories.

4 Methodology

The system consists of two procedures: one for direct and one for indirect speech. We developed a simple pattern matching algorithm for extracting the subject of sentences with direct speech. First we extract all sentences containing quoted speech (leaving out sentences that consist solely of quoted speech). Using the output of the chunking module of Frog, the system then searches for the nearest head of a noun phrase that is part of one of the following patterns:

(5) `QUOTE VP NP` (*"You, our king", cried the birds angrily.*)

(6) `VP NP QUOTE` (*"Oh dear", said the queen, "only look at my combs!"*)[6]

(7) `NP VP QUOTE` (*The fox asked, "How many tricks do you know?"*)

For indirect speech we extract the subject of matrix clauses of complementation constructions. We first locate all instances of subordinate conjunctions and interrogatives that have a verb complement dependency relation. We then extract the subject of the complement of the verb. We only included subjects that constitute a definite or indefinite noun phrase. Some examples are:

(8) *The queen wondered whether she would ever see her husband again.*

(9) *That the dragon was closing in, was obvious to the soldier.*

By applying both procedures to all texts we extract a set of actors for each text.

The set of actors should be ranked according to the actors' importance to the story. Regular term weight measures such as TF×IDF are not appropriate for this

[4] Theo Meder and Marianne van Zuijlen

[5] `http://ilk.uvt.nl/frog/`

[6] Note that this example also matches with the first rule. Unlike in English, Dutch uses inversion in subordinate clauses in which case the rule makes more sense.

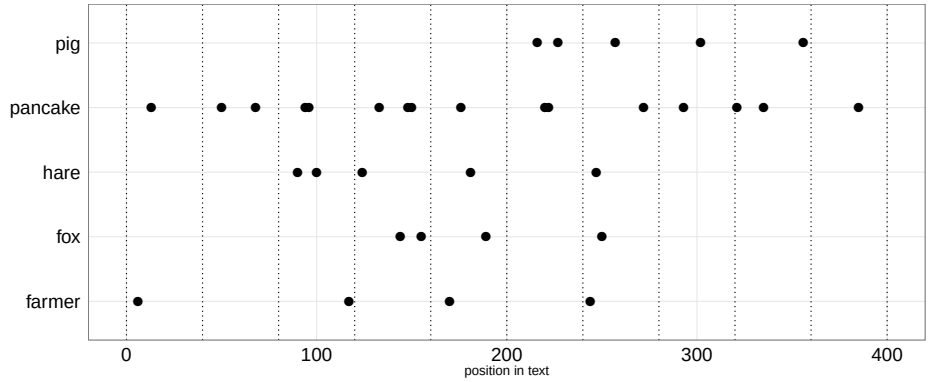


Figure 1: Lexical dispersion plot of actors in a version of *The Fleeing Pancake*.

ranking, because they estimate the importance of a term relative to the other texts in the corpus, whereas we would like to have some intra-textual measure of importance. At first sight, frequency of occurrence does not seem to be a good measure of importance either, because some actors might be referred to with high frequency in only a short text span and may not be of overall importance to the story.

We therefore propose a ranking method that is based on the dispersion of actors over a story. The basic idea is that more important actors are expected to appear at more places in the story and are more evenly distributed over the story than less important actors. Figure 1 plots the occurrences of the five actors in a version of *The Fleeing Pancake*. The pancake is present throughout the entire text whereas the fox, for example, is only briefly visible. Hence, we expect that the fox plays a less prominent role than the pancake.

We chose to use Juilland *et al.*'s [8] D statistical coefficient, because it has been reported as the most reliable dispersion measure [9, 190-191]. Juilland's D is calculated as:

$$D = 1 - \frac{V}{\sqrt{n-1}} \tag{10}$$

where n is the number of equally sized chunks in a text and V is the variation coefficient given by:[7]

$$V = \frac{\sigma}{\bar{v}} \tag{11}$$

where $\bar{v}$ is the mean frequency of a word in the different chunks and σ is the standard deviation of the frequencies in the chunks given by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(v_i - \bar{v})^2}{n-1}} \tag{12}$$

[7]The parameter n has to be set manually. In our experiments we observed little effect of the parameter within the interval $[5, 50]$ and set its value to $n = 20$.

rank	ground truth	subject baseline	our system
1	(Janneman, Jan, ventje 'little man')	Janneman	Janneman
2	(heks 'witch', wijf 'hag')	zak 'bag'	heks 'witch'
3	(tuinman 'gardener', haagknipper 'hedge cutter'), (arbeider 'worker'), (slootgraver 'ditch digger', man)	heks 'witch'	kat 'cat'
4	(kat 'cat')	huisje 'small house'	slootgraver 'ditch digger'
5	-	winkel 'store'	wijf 'hag'
6	-	pond 'pound'	haagknipper 'hedge cutter'
7	-	deur 'door'	arbeider 'worker'
8	-	kat 'cat'	-
9	-	neen 'no'	-

Table 1: Expert ranking and the results of the subject baseline (only the first 9 out of 33 results are shown) and our system for the story *Janneman in het papieren huisje* ('The Devil (Witch) Carries the Hero Home in a Sack'). Alternative names in the ground truth that refer to the same entity are placed between brackets.

Its values range from 0 (most uneven distribution) to 1 (most even distribution of a word across all chunks). To give an impression of some of the values different words can take, consider once more Figure 1. The vertical dotted lines mark the chunk boundaries at $n = 10$. Both the *farmer* and the *fox* are referred to four times, but because the farmer is more evenly distributed over the text it has a higher dispersion value, $D = 0.53$ and $D = 0.38$, respectively.

5 Experiment

For all documents in our corpus we extract a ranked list of actors. In the ideal case, the top of these lists contain the actors of a story. The extent to which this is the case reflects how well relevant items are found by our method and how well they are ranked according to the dispersion coefficient D. The results are evaluated by means of *Mean Average Precision* (MAP).

The subject of a sentence is often an entity that performs an action. Hence, subjects typically have a high chance of being an actor. We therefore compare our approach to a baseline in which all nouns and proper names are tagged as actors if they are classified by the dependency parser of Frog to hold a subject relation.

As an example, Table 1 presents the expert ranking, the results of the subject baseline (only the first 9 out of 33 results are shown) and the system for the story *Janneman in het papieren huisje* ('The Devil (Witch) Carries the Hero Home in a Sack'). Alternative names in the ground truth that refer to the same entity are

	subject baseline		system	
	complete	top three	complete	top three
frequency ranking	.739	.854	.909	.895
dispersion ranking	.792	.878	.918	.9

Table 2: MAP for both the subject baseline and the system using the frequency count ranking method and the dispersion based method. Scores are given for the complete result lists and for the top three items in the result lists against the first three ranks in the annotations.

		subject baseline		system	
		dispersion	frequency	dispersion	frequency
subject baseline	dispersion	–	$p < .0001$	$p < .0001$	$p < .0001$
	frequency		–	$p < .0001$	$p < .0001$
system	dispersion			–	$p < .03$
	frequency				–

Table 3: Pairwise Wilcoxon Signed-Rank Test using the dispersion and frequency ranking method for both the subject baseline model and our system.

placed between brackets. Since the annotations make no distinction between a canonical name of an actor and variants of that name, all alternative names are included in the set of relevant names. As noted in Section 3, we did not attempt to resolve coreferences. We thus make the simplifying assumption that all names in the result lists are to be considered as unique entities. The *Average Precision* (AP) for the baseline is .83. The system outperforms the baseline with a perfect score of 1.0, because it has identified solely relevant items.

We compare the proposed ranking method of actors to a method in which retrieved items are ranked according to their frequency of occurrence. The hypothesis is that the dispersion method should do a better job in distinguishing important from less important actors, because they are more important throughout the story and not only in small text spans.

Table 2 gives a summary of the results. The subject baseline model performs fairly well with a MAP of .739 using frequency ranking and .792 using dispersion ranking. Our system outperforms the baseline markedly with a score of .918. For our system, there is a minor difference between the two ranking methods.

To test whether the performance differences between the models are significant, we computed for all four models the AP for each document in the corpus. We then performed pairwise two-sided Wilcoxon Signed-Rank tests of which the resulting p values can be found in Table 3. All performance differences are sig-

		subject baseline		system	
		dispersion	frequency	dispersion	frequency
subject baseline	dispersion	–	$p > .2$	$p > .1$	$p > .2$
	frequency		–	$p > .05$	$p > .1$
system	dispersion			–	–
	frequency				–

Table 4: Pairwise Wilcoxon Signed-Rank Test using the dispersion and frequency ranking method for both the subject baseline and our system. AP is computed for the top three items in the result lists against the top three ranks in the annotations.

nificant. The significant differences between the two ranking methods support the hypothesis that dispersion is a helpful means to distinguish important actors from more peripheral actors.

Although the MAP shows that the system is well capable of retrieving mostly relevant items in the top retrieved results, it does not tell us whether the first, say, three items are the most important actors in a story. Recall that the annotations of the stories include a ranked list of actors. This allows us to evaluate whether the top retrieved results are not only relevant but contain the most important actors – as conceived by the annotators – as well.

The cast of a typical story contains only a few main actors and a range of supporting actors. It has one or two heroes, an opponent of the hero and either a wanted object or a helper of the hero. We recalculate the MAP scores using a constrained set of possible relevant actors in which only the first three ranks in the annotations are included. Using this set of relevant items, we compute the precision for the three highest ranked actors in the result lists (see Table 2). Again the system outperforms the baseline, albeit marginally, with a MAP of .895 using frequency ranking and .9 using the dispersion method and .878 for the baseline model using dispersion ranking and .854 using frequency ranking.

We performed the same Wilcoxon Signed-Rank tests as before to test for significance. Table 4 presents the results. We see that there are some small effects, but none of the performance differences are significant. This is interesting, because for the complete result lists, we saw significant differences between all combinations of systems (see Table 3). This suggests that the performance differences concentrate on the ranking of less important actors. It is also in this scope that the dispersion of a term is an effective indicator if its importance.

6 Conclusion

The approach taken in this paper to extract the cast from fictional texts proves to be successful. In line with literary scholars our results support the idea that

intentionality serves as a strong feature to identify actors. The intentionality is reflected in the use of two related but structurally different linguistic constructions: direct and indirect speech. By extracting these constructions from a text we obtain a high-precision method for retrieving the cast of a story. The system performs significantly better than the baseline model that marks all nouns in subject positions as cast members.

Besides the mere identification of actors, we proposed a ranking method of the actors based on the dispersion of actors over a text. The hypothesis was that actors that are more evenly distributed throughout the text are more important than actors that only appear in short text spans. The results showed that ranking by means of dispersion gives a significant performance gain over 'simple' frequency counts.

We focus our recommendations for future research on two points. First, the system proposed in this paper depends heavily on the presence of direct and indirect speech. However, neither is necessarily present in every text. To make the system more robust, it should be able to deal with these texts as well. A good starting point to improve the stability is to use the output of the present system to learn about contexts other than direct and indirect speech in which actors appear.

We made the simplifying assumption that all retrieved items are unique entities, whereas in fact they corefer to a small subset of entities. Our second point of future work is directed towards resolving these coreferences, which will require adaptation of existing coreference resolution models to the domain of fictional texts.

7 Acknowledgments

The work on which this paper is based has been supported by the Computational Humanities Programme of the Royal Netherlands Academy of Arts and Sciences, as part of the Tunes & Tales project.

References

[1] Mieke Bal. *Narratology, Introduction to the Theory of Narrative*. University of Toronto Press, Toronto Buffalo London, third edition, 2009.

[2] Thierry Declerck, Nikolina Koleva, and Hans-Ulrich Krieger. Ontology-based incremental annotation of characters in folktales. In *Proceedings of the 6th EACL Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 30–34, 2012.

[3] David Elson and Kathleen McKeown. Automatic attribution of quoted speech in literary narrative. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1013–1019, 2010.

[4] Kevin Glass and Shaun Bangay. A naïve, salience-based method for speaker identification in fiction books. In *Proceedings of the 18th Annual Symposium*

of the Pattern Recognition Association of South Africa (PRASA '07), pages 1–6, 2007.

[5] Hui-Ngo Goh, Lay-Ki Soon, and Su-Cheng Haw. Automatic identification of protagonist in fairy tales using verb. In Pang-Ning Tan, Sanjay Chawla, Chin Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7302 of *Lecture Notes in Computer Science*, pages 395–406. Springer Berlin / Heidelberg, 2012.

[6] Algirdas Julien Greimas. *Sémantique structurale*. Larousse, Paris, 1966.

[7] Walter Haeseryn. *Algemene Nederlandse Spraakkunst*. Groningen/Deurne: Marinus Nijhoff/Wolters Plantyn, 1997.

[8] Alphonse G. Juilland, Dorothy R. Brodin, and Catherine Davidovitch. *Frequency Dictionary of French Words*. Mouton de Gruyter, 1970.

[9] Michael P. Oakes. *Statistics for Corpus Linguistics*. Edinburgh University Press, Edinburgh, 1998.

[10] Tim O'Keefe, Silvia Pareti, James Curran, Irena Koprinska, and Matthew Honnibal. A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799, 2012.

[11] Josef Ruppenhofer, Caroline Sporleder, and Fabian Shirokov. Speaker attribution in cabinet protocols. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation LREC10*, pages 2510–2515. European Language Resources Association (ELRA), 2010.

[12] Jacques Rudolf Willem Sinninghe. *Volkssprookjes uit Nederland en Vlaanderen*. Kruseman, Den Haag, 1978.

[13] Karina Van Dalen-Oskam. Names in novels: an experiment in computational stylistics. *Literary and Linguistic Computing*, 2012.

[14] Antal Van den Bosch, Bertjan Busser, Walter Daelemans, and Sander Canisius. An efficient memory-based morphosyntactic tagger and parser for Dutch. In F. Van Eynde, P. Dirix, I. Schuurman, and V. Vandeghinste, editors, *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting*, pages 99–114, Leuven, Belgium, 2007.

[15] Arie Verhagen. *Constructions of Intersubjectivity, Discourse, Syntax and Cognition*. Oxford Linguistics. Oxford University Press, Oxford New York, 2005.

Bulgarian-English Sentence- and Clause-Aligned Corpus

Svetla Koeva, Borislav Rizov, Ekaterina Tarpomanova,
Tsvetana Dimitrova, Rositsa Dekova, Ivelina Stoyanova,
Svetlozara Leseva, Hristina Kukova, Angel Genov

Department of Computational Linguistics,
Institute for Bulgarian Language, BAS
52 Shipchenski Prohod Blvd., 1113 Sofia, Bulgaria
E-mail: {svetla, boby, katja, cvetana}@dcl.bas.bg,
{rosdek, iva, zarka, hristina, angel}@dcl.bas.bg

Abstract

The paper presents the partially automatically annotated and fully manually validated Bulgarian-English Sentence- and Clause-Aligned Corpus. The discussion covers the motivation behind the corpus development, the structure and content of the corpus, illustrated by statistical data, the segmentation and alignment strategy and the tools used in the corpus processing. The paper sketches the principles of clause annotation adopted in the creation of the corpus and addresses some issues related to interlingual asymmetry. The paper concludes with an outline of some applications of the corpus in the field of computational linguistics.

1 Introduction and motivation

Although parallel texts can be aligned at various levels (word, phrase, clause, sentence), clause alignment has proved to have advantages over sentence and word alignment in certain NLP tasks. Due to the fact that many of the challenges encountered in parallel text processing are related to (i) sentence length and complexity, (ii) the number of clauses in a sentence and (iii) their relative order, clause segmentation and alignment can significantly help in handling them. This observation is based on the linguistic fact that differences in word order and phrase structure across languages are better captured and formalised at clause level rather than at sentence level. As a result, monolingual and parallel text processing at clause level facilitates the automatic linguistic analysis, parsing, translation, and other NLP tasks.

Consequently, this strand of research has incited growing interest with regard to machine translation (MT). Clause-aligned corpora have been successfully em-

ployed in the training of models based on clause-to-clause translation and clause reordering in Statistical Machine Translation (SMT) – see [1] for syntax-based German-to-English SMT; [9] for English-to-Japanese phrase-based SMT; [2] for Japanese-to-English SMT; [8] for English-Hindi SMT, among others. Clause alignment has also been suggested for translation equivalent extraction within the example-based machine translation framework [7].

The Bulgarian-English Sentence- and Clause-Aligned Corpus (BulEnAC) was created as a training and evaluation data set for automatic clause alignment in the task of exploring the effect of clause reordering on the performance of SMT [6].

The paper is organised as follows. Section 2 describes the structure, content and format of the BulEnAC and the annotation tool. Section 3 summarises the approach to sentence identification and alignment. Section 4 outlines the approach to clause splitting and alignment followed by a discussion on the principles of clause annotation. Section 5 addresses the possible applications of the corpus.

2 Structure of the BulEnAC

2.1 Basic structure

The BulEnAC is an excerpt from the Bulgarian-English Parallel Corpus – a part of the Bulgarian National Corpus (BulNC) of approximately 280.8 million tokens and 8.2 million sentences for Bulgarian and 283.1 million tokens and 8.9 million sentences for English. The Bulgarian-English Parallel Corpus has been processed at several levels: tokenisation, sentence splitting, lemmatisation. The processing has been performed using the Bulgarian language processing chain [5] for the Bulgarian part and Apache OpenNLP[1] with pre-trained modules for the English part[2].

The BulEnAC consists of 366,865 tokens altogether. The Bulgarian texts comprise 176,397 tokens in 14,667 sentences, with average sentence length 12.02 words. The English part totals 190,468 tokens and 15,718 sentences (12.11 words per sentence). The number of clauses in a sentence averages 1.67 for Bulgarian compared with 1.85 clauses per sentence for English.

The text samples are distributed in five broad categories, called 'styles'. A style is a general complex text category that combines the notions of register, mode, and discourse and describes the intrinsic characteristics of texts in relation to the external, sociolinguistic factors, such as the function of the communication act.

Clause-aligned corpora typically contain a limited number of sentences and cover a particular style, domain or genre[3], such as biomedical texts [3], legal texts [4], etc.

[1] `http://opennlp.apache.org/`

[2] The OpenNLP implementations used in the development of the BulEnAC were made by Ivelina Stoyanova.

[3] The further subdivision of the styles includes categorisation into domains (e.g., Administrative: Economy, Law, etc.) and genres (e.g., Fiction: novel, poem, etc.).

The goal in creating the corpus was to cover diverse styles so as to be able to make judgments on the performance of the alignment methods across different text types. As a result, the corpus consists of the following categories: Administrative texts (20.5%), Fiction (21.35%), Journalistic texts (37.13%), Science (11.16%) and Informal/Fiction (9.84%). Figure 1 shows a comparison of the average sentence length across styles for the two languages.

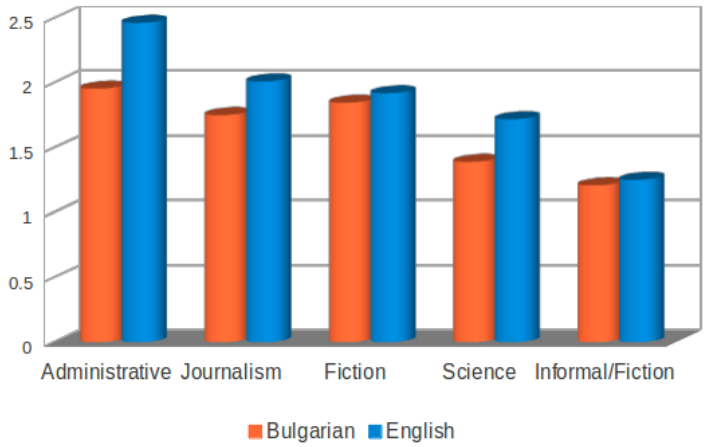


Figure 1: Average length of Bulgarian and English sentences (in terms of number of clauses) across the different styles.

2.2 Format of the Corpus

The files of the corpus are stored in a flat XML format. The words in the text are represented as a sequence of XML elements of the type `word`. Each `word` element is defined by a set of attributes that correspond to different annotation levels:

1. Lexical level (lemmatisation) – the attributes `w` and `l` denote the word form and the lemma, respectively.

2. Syntactic (sentence level) – the combination of two attributes, `e=True` and `sen=senID`, denotes the end of each sentence and the corresponding id of the sentence in the corpus.

3. Syntactic (clause level) – the attribute `cl` corresponds to the id of the clause in which the word occurs.

4. Syntactic (applied only to conjunctions) – the attribute `cl2` is used for conjunctions and other words and phrases that connect two clauses[4], and denotes the id of the clause to which the current clause is connected. The attribute `m`

[4]For brevity and simplicity such words and phrases are also termed 'conjunctions'.

defines the type of the relation between the two clauses `cl` and `cl2` (coordination or subordination), the direction of the relation (in the case of subordination) and the position of the conjunction with respect to the clauses. The inter-clausal relations are discussed in more detail in Section 4.2.

5. Alignment – the attributes `sen_al` and `cl_al` define sentence and clause alignment, respectively. Corresponding sentences/clauses in the two parallel texts are assigned the same id.

Example (1) shows the basic format of the corpus files.

Example 1 *The EU says Romania needs reforms.*

```
<word cl="864" cl_al="6c8f" l="the" w="The"/>
<word cl="864" l="eu" w="EU"/>
<word cl="864" l="say" w="says"/>
<word cl="865" cl2="864" cl_al="19f" l="PUNCT" m="N_S" w="===="/>
<word cl="865" l="Romania" w="Romania"/>
<word cl="865" l="need" w="needs"/>
<word cl="865" e="True" l="reform" sen="bc90" w="reforms.}"/>
```

Empty words (`w="===="`) are artificial elements introduced at the beginning of a new clause when the conjunction is not explicit or the clauses are connected by means of a punctuation mark. For simplicity of annotation punctuation marks are not identified as independent tokens but are attached to the preceding token.

The flat XML format is more suitable for the representation of discontinuous clauses than a hierarchical one; at the same time it is powerful enough to represent the annotation and to encode the syntactic hierarchy between pairs of clauses through the clause relation type.

2.3 The Annotation Tool

The manual sentence and clause alignment, as well as the verification and post-editing of the automatically performed alignment were carried out with a specially designed tool – ClauseChooser[5]. It supports two kinds of operating modes: a monolingual one intended for manual editing and annotation of each part of the parallel corpus, and a multilingual one that allows annotators to align the parallel units.

The monolingual mode includes: (i) sentence splitting; (ii) clause splitting; (iii) correction of wrong splitting (merging of split sentences/clauses); (iv) annotation of conjunctions; and (v) identification of the type of relation between pairs of connected clauses. Figure 2 shows the monolingual mode of ClauseChooser used for sentence and clause segmentation and annotation of clause relations. After having been segmented in the bottom left pane, the clauses are listed to the right. The type

[5]ClauseChooser was developed at the Department of Computational Linguistics by Borislav Rizov.

of relation for each pair of syntactically linked clauses is selected with the grey buttons N_N, N_S, etc.

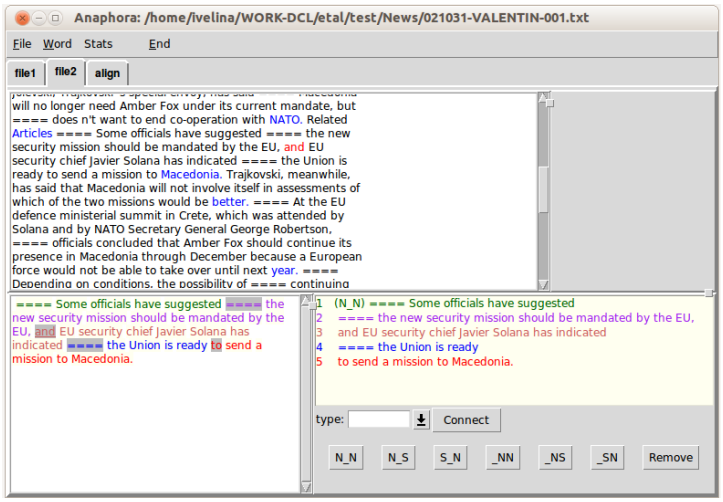


Figure 2: View of the monolingual mode of ClauseChooser

The multilingual mode uses the output of the monolingual sentence and clause splitting and supports: (i) manual sentence alignment; (ii) manual clause alignment.

3 Sentence segmentation and alignment

Both the Bulgarian and the English parts of the corpus were automatically sentence-split and sentence-aligned. The sentence segmentation of the Bulgarian part was performed with the BG Sentence Splitter. The tool identifies the sentence boundaries in a raw Bulgarian text using regular rules and a lexicon [5]. The English part was sentence-split using an implementation of an OpenNLP[6] pre-trained model. Sentence alignment was carried out automatically using HunAlign[7], and manually verified by experts.

The dominant sentence alignment pattern is 1:1 that stands for one-to-one correspondences in the two languages. The 0:1 and 1:0 alignments designate that a sentence in one of the languages is either not translated, or is merged with another sentence. Table 1 shows the distribution of the sentences in the corpus across alignment types. The category 'other' covers models with low frequency, such as 1:3, 3:1, 2:2, etc.

[6] http://opennlp.apache.org/

[7] http://mokk.bme.hu/resources/hunalign/

BG:EN alignment	frequency	in % of all
0:1	1187	7.60
1:0	225	1.44
1:1	13697	87.74
1:2	264	1.69
2:1	187	1.20
other	15	0.33

Table 1: Sentence alignment categories

4 Clause segmentation and alignment

A pre-trained OpenNLP parser[8] was used to determine the clause boundaries in the English part, followed by manual expert post-editing. The Bulgarian sentences were split into clauses manually. Clause segmentation is a language-dependent task that should be performed in compliance with the specific syntactic rules and the established grammar tradition and annotation practices for the respective languages. This approach ensures the authenticity of the annotation decisions and helps in outlining actual language-specific issues of multilingual alignment.

4.1 Clause alignment

After clause segmentation took place, the parallel clauses in the English and the Bulgarian texts were manually aligned. Alignment was performed only between clauses located within pairs of corresponding sentences.

The prevalent alignment pattern for clauses is also 1:1. However, due to some distinct syntactic properties of the languages involved, the different lexical choices, 'information packaging' patterns, etc., various asymmetries arise. The non-straightforward alignments have proved to be considerably more pronounced at clause than at sentence level as reflected in the higher frequency of clause alignment patterns of the type 1:0, 1:N and N:M (N, M>1), and the greater number of patterns that are represented by a considerable number of instances (Table 2).

1:0 and 0:1 alignments are found where a clause in one language does not have a correspondence in the other. For instance, in Example (2) the clause *he said* (2a) is not translated to Bulgarian (2b)[9].

Example 2

(a) [*La Guardia, step on it!*], [***he said.***]

[8] http://opennlp.apache.org/

[9] The Bulgarian examples are transliterated and glossed. We adopted word-by-word glossing with the following abbreviations (cf. Leipzig Glossing Rules, http://www.eva.mpg.de/lingua/pdf/LGR08.02.05.pdf): N – noun; ADJ – adjective; ADV – adverb; PTCP – participle; PST – past; PRS – present; SG – singular; PL – plural; ACC – accusative; COMP – comparative; DEF – definite.

(b) [*La Guardia, po-barzo!*]
[*La Guardia, quick-ADV;COMP!*]

1:N, N:1 patterns (N>1) stand for alignments where a given clause corresponds to a complex of (two or more) clauses. A systemic asymmetry is represented by the participial *-ing* and *-ed* clauses in English – clause 2 in (3a), and their Bulgarian counterparts. Bulgarian lacks non-finite clauses, therefore syntactic units that are headed by non-finite verbs are treated as participial constructions (the bold face part of the sentence in (3b)). In Example (3), the different clause structure of the English and the Bulgarian sentences leads to 2:1 alignment.

Example 3

*(a) [1 The Ministry announced a redistribution of financing,] [2 ==== **shifting funds to private sector projects.**]*

(b) [*1 Ministerstvoto obyavi prerazpredelenie na*
Ministry-the;DEF announce-PST;SG redistribution-N;SG of

finansiraneto, ***prehvarlyayki fondovete kam***
financing-the;N;DEF, shifting-PTCP fund-the;PL;DEF to

proekti v chastniya sektor.]
project-PL in private-the;DEF sector.]

Another frequent pattern is illustrated in Example (4). The two subordinate clauses marked in the sentence as clauses 2 and 3 in (4a), are translated as prepositional phrases PP_2 and PP_3, respectively[10]. As a result, the Bulgarian translation of the 3-clause English sentence consists of a single clause (4b); hence the alignment pattern is 3:1.

Example 4

*(a) [1 This Regulation does not go beyond] [2 **what is necessary**] [3 **to achieve those objectives.**]*

(b) Nastoyasthiyat reglament ne otiva po-dalech
Present-the;DEF regulation not go-PST;SG beyond-ADV;COMP

(PP_2 *ot neobhodimoto (*PP_3 *za postigane na*
from necessary-the;DEF for achievement-N of

tezi tseli)).
this-PL objective-PL.

Alignments of the type N:M (N,M>1) represent complex-to-complex correspondence and are relatively rare (0.84% of the clauses, Table 2). Example (5) illustrates an alignment pattern of the type 3:2. The English matrix clause 1 in (5a)

[10]The phrase labels are given for expository purposes. The clause-aligned corpus does not include annotation of phrasal categories.

is translated into Bulgarian (5b) by means of clause 1 and the part of clause 2 in boldface. The object of the English clause 1 *measures* (BG: merki) is the subject of the Bulgarian subordinate clause 2 *da badat vzeti merki...* (EN: for measures to be taken...) that roughly corresponds to the prepositional phrase in the English counterpart *for measures*. On the other hand, the subordinate clauses 2 and 3 in the English sentence are rendered as the prepositional phrase PP in Bulgarian (5b).

Example 5

*(a) [1 He urged **for measures**] [2 to help displaced persons] [3 return to their homes.]*

(b) [*1 Toy nastoya*] [*2* ***da badat vzeti***
He insist-PST;SG to be-PRS;PL take-PTCP;PL

merki *(PP za podpomagane na zavrashtaneto*
measure-PL for help-N of returning-the;N;DEF

na prinuditelno izselenite po tehnite domove).]
of forcefully displaced-PTCP;DEF;PL to their home-PL.

The distribution of the alignment pairs is given in Table 2.

BG:EN alignment	frequency	in % of all
0:1	1745	7.05
1:0	482	1.95
1:1	18997	76.80
1:2	2256	9.12
1:3	239	1.33
1:4	99	0.40
2:1	621	2.51
2:2	87	0.32
other	128	0.52

Table 2: Clause alignment categories.

Non-straightforward alignment patterns account for considerable number of 0:1 (7.05%) and 1:2 (9.12%) clause alignments in Bulgarian-English, with the reverse types amounting to just 1.95% (1:0) and 2.51% (2:1), respectively. These results suggest that a stronger tendency exists for 1:N (N>1) correspondences for Bulgarian-to-English than for English-to-Bulgarian. Some of the factors for this trend include the different segmentation into clauses as in the case of participial constructions versus participial clauses, and the rendition of prepositional phrases as clauses or vice versa.

4.2 Annotation of clause relations

The BulEnAC is supplied with partial syntactic annotation that includes:

(i) delimiting the sentence and clause boundaries;

(ii) identifying the type of relation (subordination or coordination) between the clauses in a sentence;

(iii) identifying the linguistic markers that introduce clauses – conjunctions, adverbs, pronouns, punctuation marks, etc.

A clause relation is defined between a pair of clauses. We were interested in the type of relation between the clauses, the ordering of clauses that stand in a given relation, the position of the conjunction, and language-specific clause-to-clause ordering constraints. With respect to the relation each clause in the pair is identified as either *main* or *subordinate* with at least one being *main*. In this paper the term *main* is used in a broader sense that encompasses both the meaning of an independent clause and that of a superordinate clause. Thus, *main* (N) denotes either a clause with equal status as the other member of the pair or one that is superordinate to it. *Subordinate* (S) status is assigned to a clause that is syntactically subordinate to the other member of the pair.

The status of the clauses is defined with respect to a particular clause relation and is therefore relative. Consequently, the relationship between a pair of coordinated independent or coordinated subordinate clauses is both N_N, cf. Example (6) for independent and Example (7) for dependent clauses. In the case of coordinated subordinate clauses, the dependent status of the pair is denoted by the relation N_S established between their superordinate and the first of the subordinate clauses (7b).

Example 6

*(a) [*N1 *I usually forget things,] [*N2 ***but***$_{\text{N1\_N2}}$ *I remembered it!]*

*(b) [*N1 *He asked her] [*S ***if***$_{\text{N1\_S}}$ *he could pick her up on the morning of the experiment] [*N2 ***and***$_{\text{N1\_N2}}$ *she agreed gratefully.]*

Example 7

(a) [1 Dutch police authorities said] [2 ***they were illegal immigrants****] [3* ***and would be deported.****]*

(b) [1 N *Dutch police authorities said] [2* S **====**$_{\text{N\_S}}$ *they were illegal immigrants]*

(c) [2 N1 *they were illegal immigrants] [3* N2 ***and***$_{\text{N1\_N2}}$ *would be deported.]*

A syntactically subordinate clause that is superordinate to another clause has the status *main* with respect to it. For instance, in (8a) clause 2 is subordinate to the matrix clause – clause 1 (8b), and a main clause with respect to clause 3 (8c):

Example 8

(a) [1 This Regulation does not go beyond] [2 ***what is necessary****] [3* ***to achieve those objectives.****]*

(b) *[1 N This Regulation does not go beyond] [2 S* ***what***$_{\texttt{N\_S}}$ *is necessary]*

(c) *[2 N ...what is necessary] [3 S* ***to***$_{\texttt{N\_S}}$ *achieve those objectives.]*

In the languages under consideration the following three clause ordering models cover almost all the cases: N_N, N_S and _SN.

4.3 More on translational asymmetries

Translational asymmetries stem also from different information distribution, lexical and grammatical choices, reordering of the clauses with respect to each other and (cross-clause boundary) reordering of constituents. In this section, we point out two types of asymmetry concerning the internal structure of clauses and their relative order within the sentence.

A frequent pattern found in the corpus is the selection of verbs with different types of complements motivated by grammatical structure, lexical choice or other factors. In the aligned sentences in Example (9) the choice of the Bulgarian verb *nastoyavam* (insist) as the translation equivalent of the English object-control verb *urge* predetermines the difference in the structure of the matrix and the subordinate clause in the two languages – in (9a) *Croatia* is the object of the main clause, whereas its counterpart *Harvatska* is the subject of the subordinate clause in (9b).

Example 9

(a) *[N European Parliament urges* ***Croatia****] [S to fully cooperate with the Tribunal.]*

(b)	[N	*Evropeyskiyat*	*parlament*	*nastoyava*]	[S	***Harvatska***
		European-the;DEF	*Parliament*	*insist-PRS;SG*		*Croatia*

da	*satrudnitchi*	*napalno*	*na*	*tribunala.*
to	*cooperate-PRS;SG*	*fully*	*to*	*tribunal-the;DEF.*

Another frequent example is the different order of the clauses in a sentence. For instance, in Example (10), the English clauses N_S (10a) are in reverse order as compared with the Bulgarian translation – _SN (10b).

Example 10

(a) *[N She had to make a detour] [S* ***to*** *get to the stove.]*

(b)	[S	***Za da***	*stigne*	*do*	*pechkata,*]
		In order to	*get-PRS;SG*	*to*	*stove-the;DEF*

[N	*tya*	*tryabvashe*	*da*	*mine*	*pokray*	*tyah.*]
	she	*must-PST;SG*	*to*	*go-PRS;SG*	*past*	*they-ACC;PL.*

Translation asymmetries represent a systemic phenomenon and account for the inter-lingual variations in grammatical structure, lexicalisation patterns, etc. At the

same time, they often give rise to wrong alignments, mistranslations, and other errors. Therefore, the successful identification of such phenomena and their proper description and treatment is a prerequisite for improving the accuracy of alignment and translation models.

5 Conclusion and applications

The development of the Bulgarian-English Sentence- and Clause-Aligned Corpus is a considerable advance towards establishing a general framework for syntactic annotation and multilingual alignment, as well as for building significantly larger parallel annotated corpora. The manual annotation and/or validation has ensured the high quality of the corpus annotation and has made it applicable as a training resource for various NLP tasks. As the goal was to explore the influence of clause alignment, further levels of alignment were only partially attempted as a technique enhancing the alignment method.

The quality of the manual clause splitting, relation type annotation and alignment was guaranteed by inter-annotator agreement. Each annotator made at least two passes of each Bulgarian and English file, one performed after the final revision of the annotation conventions. Clause segmentation was additionally validated at the stage of clause alignment.

The NLP applications of the BulEnAC encompass at least three interrelated areas: (i) developing methods for automatic clause splitting and alignment; (ii) developing methods for clause reordering to improve the training data for SMT [6]; (iii) word and phrase alignment. These lines of research will facilitate the creation of large-scale syntactically and semantically annotated corpora. In the field of the humanities the corpus is a valuable resource for studies in lexical semantics, comparative syntax, translation studies, language learning, cross-linguistic studies.

The BulEnAC will be made accessible to the scholarly community through the unified multilingual search interface of the Bulgarian National Corpus[11].

6 Acknowledgements

The present paper was prepared within the project *Integrating New Practices and Knowledge in Undergraduate and Graduate Courses in Computational Linguistics* (BG051PO001-3.3.06-0022) implemented with the financial support of the *Human Resources Development* Operational Programme 2007-2013 co-financed by the European Social Fund of the European Union. The Institute for Bulgarian Language takes full responsibility for the content of the present paper and under no conditions can the conclusions made in it be considered an official position of the European Union or the Ministry of Education, Youth and Science of the Republic of Bulgaria.

[11] `http://search.dcl.bas.bg`

References

[1] B. Cowan, I. Kucerova, and M. Collins. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Sydney*, pages 232–241, 2006.

[2] C.-L. Goh, T. Onishi, and E. Sumita. Rule-based reordering constraints for phrase-based SMT. In *Proceedings of the 15th International Conference of the European Association for MT, May 2011*, pages 113–120, 2011.

[3] J.-D. Kim, T. Ohta, and J. Tsujii. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(10), 2008.

[4] C. Kit, J.J. Webster, K. Kui Sin, Pan H., and H. Li. Clause alignment for bilingual Hong Kong legal texts: A lexical-based approach. *International Journal of Corpus Linguistics*, 9(1):29–51, 2004.

[5] S. Koeva and A. Genov. Bulgarian language processing chain. In *Proceedings of Integration of Multilingual Resources and Tools in Web Applications. Workshop in conjunction with GSCL 2011, University of Hamburg*, 2011.

[6] S. Koeva, B. Rizov, E. Tarpomanova, Ts. Dimitrova, R. Dekova, I. Stoyanova, S. Leseva, H. Kukova, and A. Genov. Application of clause alignment for statistical machine translation. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), Korea*, 2012.

[7] S. Piperidis, H. Papageorgiou, and S. Boutsis. From sentences to words and clauses. In J. Veronis, editor, *Parallel Text Processing, Alignment and Use of Translation Corpora*, pages 117–138. Kluwer Academic Publishers, 2000.

[8] A. Ramanathan, P. Bhattacharyya, K. Visweswariah, K. Ladha, and A. Gandhe. Clause-based reordering constraints to improve statistical machine translation. In *Proceedings of the 5th International Joint Conference on NLP, Thailand, November*, pages 1351–1355, 2011.

[9] K. Sudoh, K. Duh, H. Tsukada, T. Hirao, and M. Ngata. Divide and translate: improving long distance reordering in statistical machine translation. In *Proceedings of the Joint 5th Workshop on SMT and Metrics MATR*, pages 418–427, 2010.

Cleaning up and Standardizing a Folktale Corpus for Humanities Research

Iwe Everhardus Christiaan Muiser[1], Mariët Theune[2] and Theo Meder[3]

[1]Database group, University of Twente, Enschede, the Netherlands
[2]Human Media Interaction, University of Twente, Enschede, the Netherlands
[3]Meertens Institute, Amsterdam, the Netherlands

E-mail: e.c.muiser@utwente.nl

Abstract

Recordings in the field of folk narrative have been made around the world for many decades. By digitizing and annotating these texts, they are frozen in time and are better suited for searching, sorting and performing research on. This paper describes the first steps of the process of standardization and preparation of digital folktale metadata for scientific use and improving availability of the data for humanities and, more specifically, folktale research. The Dutch Folktale Database has been used as case study but, since these problems are common in all corpora with manually created metadata, the explanation of the process is kept as general as possible.

1 Introduction

Recordings in the field of folk narrative have been made around the world for many decades. Storage, annotation and studies of these corpora by digital means however, have only begun just recently. By annotating and creating digital versions of these tales, they are virtually 'frozen' in time, which opens up a window for researchers to compare historical versions of narratives. Many types of studies can be efficiently done by searching and comparing the tales once they are put in a digital framework [1]. By adding metadata such as keywords, dates, and geographical locations to the original texts it becomes easier to search, categorize and navigate through a corpus. An additional advantage of digitization is that the information can now be shared with researchers and other interested people all over the world. This creates wonderful opportunities to analyze and compare the similarities and differences between folktales in various cultures. In general the more metadata is present, the more extensive comparative research can be performed.

For a few decades now, the Meertens Institute in Amsterdam has been collecting folktales. Since 1994 these tales are being digitized and put into the Dutch

Folktale Database (DFDB), which came online in 2004 (www.verhalenbank.nl). Over the years, the descriptive metadata assigned to these tales have undergone several transformations due to periodically changing demands. Data fields have been added to describe and identify the original texts in increasing detail, for instance with an indication of the motifs in the text, the creation and mutation date, and whether the tale is extreme in nature. Today, the database contains roughly 42.000 tales, all of which are provided with a rich set of manually added metadata. Several tens of thousands of texts are still waiting for annotation or digitization, and many circulating Dutch folktales have not even been recorded yet.

The Dutch folktale collection in the DFDB is currently being used within the FACT project[1] to investigate the automatic annotation of folktales with metadata such as genre, language and keywords. The project aims to support humanities research by developing new methods for automatic metadata extraction, classification and clustering of folktales. This is a challenging task because many of the folktale texts have been taken down within the context of oral performance, and can contain vernacular language (including laughter, pauses, hesitations, incomplete and imperfect sentences etc.), dialect, slang, and mixed languages. This often causes traditional natural language processing (NLP) methods to be insufficient.

The manually annotated metadata of the folktale corpus are a very valuable resource for the automatic annotation of folktale texts. Information about the date and place of narration is likely to be of use for automatic language identification, and in experiments on classification of folk narrative genres we have shown that metadata such as keywords, summary, and date can be used to improve performance [2]. However, for the current metadata annotations to be optimally useful for training and testing automatic classifiers, an organized metadata setup has to be present, and the number of errors and inconsistencies in the data fields has to be minimized. For example, our language identification experiments may have been hampered by inconsistent labeling of mixed language documents [3].

Since the annotation of folktales has been done by hand by about sixty different annotators over a large period of time, a fair portion of errors and deviations from the input standard can be expected. Van den Bosch et al. have observed error rates up to 5% in cultural heritage databases [4].

This paper describes the first steps of the process of standardization and preparation of the DFDB metadata for scientific use, improving availability of the data for humanities and, more specifically, folktale research. The paper focuses on the Dutch Folktale Database, but, since these problems are common in all corpora with manually created metadata, the explanation of the process is kept as general as possible. In Section 2, some light is shed on common errors in cultural heritage databases. Section 3 discusses metadata standards. The actual standardization of metadata values of our folktale corpus is described in Section 4.

[1] Folktales As Classifiable Texts, http://www.elab-oralculture.nl/fact

2 Errors and inconsistencies in cultural heritage databases

Many digitized cultural heritage collections like the DFDB contain information that was created manually. Most information has been stored in free text formatted databases, in many respects functioning as a digital encyclopedia, without taking into account the possibilities of digital analysis laying ahead. The free text input left room for freedom of annotation, comments and explanations. This is beneficial for free text searching but disastrous for ordering data and structured search. It makes browsing and visualization a very challenging task. Standardization of metadata is therefore of great importance for the field.

Manual free text input also allows mistakes and inconsistencies to easily sneak in. Van den Bosch et al. [4] discuss three types of common errors in cultural heritage databases. Items with *typing and spelling errors* are unlikely to turn up in search results. *Wrong column errors* occur when the content of database columns has been misplaced or switched. *Content errors* are usually due to wrong, or alternative, interpretations and classifications of corpus items (in our case, folktales). Fixing these errors is a step that can be made after clearing up more generic inconsistencies. In the metadata of our corpus we found the following inconsistencies:

- Deviations from a set standard. In dates, for example, 'February 1st 2012' could be '1 feb. 2012', '01-02-2012' or '2012-02-01'. Names can also have many formats like 'Jan van der Vaart', 'van der Vaart, Jan' or 'J. vd Vaart'.
- Differences in delimiters. Names can be separated by comma's, ampersands, or other characters.
- Addition of comments in divergent formats. A value can be uncertain, causing some annotators to add '?', '[?]', 'UNKNOWN', or putting the complete value between square brackets.
- Capitalization / punctuation variations. Some tale titles end with a full stop while others do not. This is also the case for other values such as tale type and geographical location. Names and geographical locations that need to start with a capital character are sometimes completely written in lowercase.

The paper focuses mainly on fixing these inconsistencies. Correction of actual errors will be addressed in a later stage of the project.

3 Metadata standards and infrastructures

Most cultural heritage databases start out using similar terms such as date, names, keywords, or geographical locations. Yet, it is common that during the database's life span, metadata terms are added and their values become more complex. To prepare for cooperation with, and to prevent bad communication between, other collections around the world [5], it is preferable to comply with standards where possible.

Standards for data and metadata are available in abundance. All these standards have their own (dis)advantages and levels of complexity. For standardization of the DFDB we choose to adopt Dublin Core[2], because it is the most basic and widely accepted standard. An additional plus is that the web-publishing platform of choice for the DFDB, Omeka[3], has Dublin Core as its primary standard. Dublin Core is a classic set of 15 metadata terms which can be used to describe a large range of media resources (web and physical). Dublin Core terms can be interpreted loosely due to their general nature. Mappings of database fields to Dublin Core terms might not always be intuitive due to differences in naming conventions. Before assigning a name, the type of data must be properly analyzed. For the sake of internationalization, the field names have to be defined in English. In case of the DFDB however, terms are defined in Dutch, and can, when literally translated, have a slightly different meaning. This can potentially cause confusion when used in an international context.

All data of the Meertens Institute, including the DFDB, will eventually be made available through the CLARIN initiative [6], which aims for a sustainable data infrastructure to aid interoperability in the humanities, and more specifically, linguistics. CLARIN uses a structured data format for metadata called Component MetaData Structure (CMDI). Datasets have to be converted to this format before they can be accessed through CLARIN. To avoid conversion problems, it is of course best to have data that do not deviate from a strict standard. Conversion tools are available for Dublin Core and other, more linguistically oriented metadata schemes such as OLAC[4] and IMDI[5]. To ensure interoperability, CLARIN makes use of ISOCAT, a framework for defining data categories that comply with the ISO/IEC 11179 family of standards. Here metadata terms and their definitions and restrictions can be registered, or existing terms can be adopted when deemed suitable. The latter is always encouraged to limit the number of terms in ISOCAT.

4 Standardization

In this section we describe the main steps involved in standardization of a corpus. First and foremost, it is important to collect the wishes of the users of the database in question. Users rely on data that can be found and sorted based on all available terms. The only way to facilitate this is to make sure that all new items conform to a strict and properly documented standard before being submitted to the database.

4.1 Dutch Folk Tale Database metadata

The DFDB encompasses a rich set of metadata fields: a total of 29 terms supplement the original text. Annotation and input of folktales in the DFDB was largely

[2] http://www.dublincore.org
[3] http://www.omeka.org
[4] http://language-archives.org
[5] http://www.mpi.nl/imdi/

done by interns and employees of the Meertens Institute. Most metadata fields have been entered in a free text format. Tables 1 and 2 show the terms of the DFDB, including mappings to their future standards.

Dublin Core term	DFDB term	Explanation
4.1. Title	title	Title of the folktale
4.2. Subject	folktale/ATU type	Folktale type code
4.3. Description	text summary	Summary of the text
4.4. Type	source format	Original source type (e.g., book, oral)
4.5. Source	text source	Description of the source
4.6. Relation	-	Empty for future use
4.7. Coverage	region	Geographical information
4.8. Creator	narrator	The person who told the tale
4.9. Publisher	-	Not used
4.10. Contributor	collector	The person that recorded the tale
4.11. Rights	copyrights	Specifies if a text is copyrighted
4.12. Date	date	The date of narration or discovery
4.13. Format	-	Empty for future use
4.14. Identifier	id number	The internal identifier code
4.15. Language	language	The language or dialect of narration

Table 1: A folk tale object's metadata terms mapped to Dublin Core terms

DFDB term (English)	Explanation
literary	Specifies if the text is literary
subgenre	Genre of the tale (fairy tale, joke, etc.)
motifs	Comma separated Thompson motif codes
keywords	Comma separated list of keywords
named entities	Named entities mentioned in the text
remarks	Additional information about the text
corpus	Corpus code
definition / description	ATU information
kloeke georeference	Kloeke georeference code of region
kloeke georeference in text	Locations mentioned in the text
extreme	Specifies if a text is extreme in nature

Table 2: List of original DFDB terms that need to be registered at ISOCAT

For a selection of fields, scripts were written to analyze, and to convert the original free text values into well formatted values.

The *date* data type is one of the most diversely composed values in this database. It ranges from perfectly composed ISO 8601 international standard (YYYY-MM-DD), to Dutch standard (DD-MM-YYYY), to completely textual values like 'Third quarter seventeenth century' and 'stumbled upon on 12 February 2003'. Some values have question marks, square brackets, commas and points in them. Sporadi-

cally the date has been supplemented with information about the era when the story was being told, or when the story took place.

Statistics about observed variations in date formats are shown in Table 3. Before the dates were checked for inconsistencies, they were lower cased and spell-corrected. Square brackets around the date number, month, year or whole date value were removed, values like 'sep.', 'sept', 'sept.', and 'september' were changed to 'm09', and day names were taken out as well. Some implausible values like 'February 30th 1969' were recognized by the scripts and manually corrected before the final conversion.

The *region* column is another value that can deviate a fair amount from any defined standard although the vast majority has a 'place (province)' composition. Multiple locations separated by several types of delimiters have been found. Spelling mistakes, alternative or historical place names and additional commentary are no exception.

Format	**Amount**	**Percentage**
DD [month] YYYY	25939	62.75%
YYYY	4909	11.88%
[part of] [century]	2345	5.67%
[month] YYYY	1170	2.83%
Easily recognized structures (above)	34363	83.13%
Other recognized structures *	5393	13.05%
Tales with no date value	1580	3.82%
Total tales	41336	100%

** Values containing enough information to compose structured date values*

Table 3: Statistics about the different date formats that were found in the DFDB.

Format	**Amount**	**Percentage**
Place (Province)	32284	78.10 %
Place name only	1466	3.55 %
Province only	1205	2.92 %
Easily recognized structures (above) *	34955	84.56 %
Other recognized structures **	952	2.30 %
Items without geographic information	5429	13.13 %
Total tales	41336	100 %

** Values containing enough information to retrieve additional geographic data*
*** Including all non-Dutch/Belgian locations and exotic formats*

Table 4: Statistics about the different geographical formats that were found in the DFDB.

An alternative way to specify a geographic reference in the DFDB is to assign a *Kloeke georeference*. This is a geographical code for the place where the story

was told. To facilitate his dialect research, in 1926 Gesinus G. Kloeke (1887-1963) divided the map of the Low Countries into a grid and added codes to (most) places. The system was long ago adopted by the Meertens institute as the geographical standard. It will remain to be supported in the future because many books, papers and publications make use of it.

The *source format* data type has always been filled using a selection list and contains abbreviated values, nicely conforming to the standard. It holds a code for the type of source from which the story originates. For instance, B stands for 'boek' (book) while M stands for 'mondeling' (oral), meaning that the story was recorded from oral transmission.

In the fields for *motifs, subgenre*, and *keywords* we see similar problems as for date and geographic location. Several types of delimiters have been used, and various ways to indicate uncertainty about the assigned values. This can make searching and separation of values problematic.

In the *type* field we found 24 values with typing errors that were easily traceable, but also 65 values that could not be found in any of the tale type indexes used by the DFDB. No controlled vocabulary of *keywords* was defined for this collection. A keyword can be a number, name or a word in any time, state, or language. After extraction of all keywords from the database, we ended up with a total of 562480 assigned keywords of which 41555 are unique. Of the assigned keywords, 993 had additional commentary, disclosing information about the context of the keyword, while 50 contained question marks or square brackets to denote uncertainty of meaning or relevance. Some (translated) examples are: 'mirror [black]', '[cannibalism]', 'piggy (?)', and 'punishment?'.

For the fields containing person names, such as *collector* or *narrator*, input conventions have been appointed but these have not always been respected. Most frequently the name is written as 'surname, first name' but often deviations like the reverse, comma-less, title plus name, or just first name have been used. In more recent items, obtained from the Internet, forum user names have been entered. Since it is not always clear which name is the first name, this is a hard problem to tackle fully automatically.

4.2 Standardization of DFDB metadata

It is possible to determine an order of importance in the standardization of the values in a cultural heritage database based on the search and browse behavior of users. The values that are most often used for search, sorting and visualization in the DFDB are all fields, title, keywords, dates and geographical location. Here we discuss the standardization of dates and geographical locations in detail, and treat the rest as standardization problems of a similar nature. With this standardization step we take into account the properties and limitations of the chosen metadata standard, Dublin Core.

Dates As explained above, the date values in the DFDB have always been entered in free text. They can range from a perfectly formatted date value to a complete sentence with comments. To improve functional searching, sorting and conversion, we need to be able to capture all this information in a simple computer readable format. Most free text date values represent either a single date or a time span. Therefore we chose to adopt a data container with a range of two dates defined as 'from', and 'up to and including', both conforming to the ISO 8601 standard. This standard defines a date as YYYY-MM-DD. If an item's date is a single date, both these dates will be the same. For existing free text values, we propose a strictly defined interpretation, determined in consultation with the humanities researchers maintaining the DFDB. This interpretation is shown in Table 5. Strict values have been determined to represent 'end of', and 'beginning of' indicators. The observed 'before', and 'middle' or 'halfway' values have also been quantified.

Description)	**Definition**
midway point century	YY51-01-01
midway point year	YYYY-06-01
midway point month X	YYYY-[X-(MAX_MM_DATE/2)]
beginning of / end of century	first/last 20 years
beginning of / end of year	first/last 2 months
beginning of / end of month	first/last 7 days
mid/halfway century	10 year window around midway point century
mid/halfway year	1 month window around midway point year
mid/halfway month	3 day window around midway point month

Table 5: Quantified definitions of free text dates. MAX_MM_DATE stands for the last day of the month in question

In the future, users will be allowed to enter a date in free text format that is supported by the definitions above. This will then be automatically translated as shown in Table 6. All structured date ranges will be placed in the Dublin Core date field after processing and approval of the annotator. Additional comments concerning a date will have to be specified in the item's comments field. The date range from the item will in turn be used to generate human readable dates like '14th century', or 'winter 2012' for viewing. A somewhat similar approach to accommodating users' preference for 'common language' over strict date formats is that of Petras et al. [7], who map named time periods (e.g., the Renaissance or the Cold War) to date ranges.

Geographical locations For the 'region' field of the narration of a folk tale, the original, and most frequently occurring format is 'place_name (province)'. The value is based on the name of the location at the time of narration. At present, it is hard to search the DFDB for tales that were recorded in a particular county or

Description	Translation
Precise date	(example) 1550-01-01 - 1550-01-01
Cth Century	[(100(C-1))+1]-01-01 - [100C]-12-31 (official)
Only YYYY available	YYYY-01-01 - YYYY-12-31
Only YYYY-MM available	YYYY-MM-01 - YYYY-MM-[MAX_MM_DATE]
Xth quarter of year YYYY	YYYY-[3X-2]-01 - YYYY-[3X]-[MAX_MM_DATE]
Xth quarter of Cth century	[(100(C-1))+1+(25X-25)]-01-01 - [(100C)+(25X)]-12-31
Beginning of Cth century	[(100(C-1)+1)]-01-01 - [(100(C-1)+20)]-12-31
Beginning of year YYYY	YYYY-01-01 - YYYY-02-31
Beginning of month MM	YYYY-MM-01 - YYYY-MM-07
End of Cth century	[(100(C-1)+81)]-01-01 - [(100(C))]-12-31
End of year YYYY	YYYY-11-01 - YYYY-12-31
End of month MM	YYYY-MM-[MAX_MM_DATE-7] - YYYY-MM-[MAX_MM_DATE]
[season] year	[Begin date season] - [end date season] (Outer possible dates of that season)

Table 6: Strictly defined interpretations of free text date values

region. When taking future international cooperation into account, it is preferable to supply higher order information such as country and continent name. A suitable hierarchical order for locations would be: Geographical coordinate (latitude, longitude), spot (building name/artwork/tree/dune), street (with optional number), place (village/city/lake), county, region (area/nature reserve/mountain), province, country, and continent (and perhaps even planetary body for future entries). Some examples:

- Full set: (53.360304, 5.214203), Brandaris, Torenstraat, West-Terschelling, Terschelling, - , Friesland, the Netherlands, Europe, Earth
- Partial set: (52.37403, 4.88969), - , - , Amsterdam, Amsterdam, - , Noord-Holland , the Netherlands, Europe, Earth
- Partial set: (51.74308, 4.77339), - , - , - , - , Biesbosch, Noord-Brabant, the Netherlands, Europe, Earth

An open source geographical database that can supply such information is Geonames.[6] This database contains roughly 8 million geographical entries worldwide, and their corresponding coordinates, in a hierarchical manner. The items already present in the DFDB will be supplemented with all available information that can be retrieved from Geonames. For future input however, Google Places[7]

[6] http://www.geonames.org/

[7] https://developers.google.com/places/

will be used for retrieval of coordinates and hierarchical geographic information. We will do this using a simple geolocation plugin for the Omeka content management system which has been extended to fit the information needs of the DFDB.

To facilitate historical annotation of a folktale, we either need to create the freedom for an annotator to supply historically sound information, or to consult a very complete spatial history database. The latter option is being investigated by several groups around the world [8, 9], yet no completely open source solutions are available at the moment. When these become available, it is still possible to adopt them. We leave room in our system for manual alterations in the data provided by Google, so that names of old towns and counties can be supplied.

We will continue to support the Kloeke georeference codes. This way, we assure a link with the other collections of the institute and are still able to use the old visualization methods used by some researchers. It will however no longer be necessary to look up this code manually, adding to the reduction of input steps for a new item. The Kloeke georeference data will not be stored in a standard Dublin Core field since it is useless for researchers outside the Meertens institute.

Tale types, motifs and keywords Since these values have always been typed, or copy pasted into place, some input errors have been made. However, only little inconsistency was observed in these values. Datasets are available for the tale types and motifs that were cross referenced for correctness. This yielded lists that were of manageable size for manual curation.

The generated list of unique keywords will be used to give suggestions by means of auto-completion when an annotator attempts to add them. This will prevent further addition of variants. In a sense, this is an ideal compromise between a controlled vocabulary and complete freedom.

4.3 Meta-metadata standardization

If an annotator is uncertain about a metadata value, it should be somehow stored in the data set. In case of automatic annotation it is desirable to assign a confidence level to the annotation. An annotator can consequently approve or reject the outcome. This type of information can be considered meta-metadata. It is currently specified for the following DFDB fields: Tale type, Narrator, Kloeke georeference (location of narration), Kloeke georeference (locations in the tale).

Meta-metadata could be stored in the following ways:

- A special character or note in the data itself, e.g. #D:[confidence level]
- An additional term or column in the database, e.g. location_disputed (yes/no)
- An additional dataset containing the value's id and status, e.g. value_id, status, status_score, date_created

A *special character* in the data itself might confuse the user of the data, and can be seen as data pollution. An *additional term* in the database is a good solution

for a “quick fix". The disadvantage is that for every metadata descriptive term a column has to be added to the database. The last option from the list is the most permanent and modifiable solution. An additional system can monitor the state of each value of each item of the database. An additional table can be created containing the item's value's id, a meta-metadata type, a score and a date. Now, an item can be flagged as disputed, or as being computer generated with a confidence interval of 0.43, or approved by a user. Table 7 shows some examples of these values.

item_value_id	meta-metadata-type	confidence	date_created
(item 1, tale type)	disputed	-	01-01-2001
(item 25, language)	generated	0.43	10-01-2010
(item 25, language)	approved	-	11-01-2010

Table 7: A few examples of rows in a meta-metadata table

4.4 Challenges

The DFDB, like many other online cultural heritage collections, is not static but is constantly being supplied with new items. Because annotators and users rely on a working system, everything should stay that way. The standardization process has to be carefully planned and prepared; trial and error will confuse the users and cause even more errors and inconsistencies. The process is best carried out in big steps. It can be compared to repairing a moving bicycle.

5 Conclusion

At first glance, the standardization of a cultural heritage database seems like a straightforward job which could be done through some simple filter actions, value separations, and regular expressions. However, the contrary is true. Many factors have to be taken into account before the actual data can be touched and changed. The data need to be studied in detail to get an overview of all the possible types of values. Standardization of data brings along a large set of problems. We have shown that manual annotation of cultural heritage databases can spawn several types of errors and inconsistencies. We have discussed the choice of metadata standards and infrastructures and how to connect with them. Existing data standards and conventions will have to be respected, or developed when none are present. The longer conventions are not followed, the larger the divergence in values becomes, and the harder the clean up operation will become. On top of this, the addition of items to the database by annotators is a continuous process which makes a faceted standardization of the database and its functions a difficult task. We hope that this paper can be a helpful asset for future endeavors of a similar kind.

6 Acknowledgments

This work has been carried out within the Folktales as Classifiable Texts (FACT) project, part of the Continuous Access To Cultural Heritage (CATCH) program funded by the Netherlands Organization for Scientific Research (NWO).

References

[1] James Abello, Peter Broadwell and Timothy R. Tangherlini. Computational Folkloristics, *Communications of the ACM* 55(7): 60–70, 2012.

[2] Dong Nguyen, Dolf Trieschnigg, Theo Meder, and Mariët Theune. Automatic Classification of Folk Narrative Genres, *Proceedings of the Workshop on Language Technology for Historical Text(s) (KONVENS 2012)*, pp. 378–382, 2012.

[3] Dolf Trieschnigg, Djoerd Hiemstra, Mariët Theune, Francisca de Jong, and Theo Meder. An Exploration of Language Identification Techniques for the Dutch Folktale Database, *Proceedings of the Workshop on Adaptation of Language Resources and Tools for Processing Cultural Heritage (LREC 2012)*, pp. 47–51, 2012.

[4] Antal van den Bosch, Marieke van Erp, and Caroline Sporleder. Making a Clean Sweep of Cultural Heritage, *IEEE Intelligent Systems* 24(2):54–63, 2009.

[5] Theo Meder. From a Dutch Folktale Database towards an International Folktale Database, *Fabula* 51(1-2): 6–22, 2010.

[6] Tamás Váradi, Steven Krauwer, Peter Wittenburg, Martin Wynne and Kimmo Koskenniemi. CLARIN: Common Language Resources and Technology Infrastructure, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, 2008.

[7] Vivien Petras, Ray R. Larson, Michael Buckland. Time Period Directories: a Metadata Infrastructure for Placing Events in Temporal and Geographic Context, *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pp. 151–160, 2006.

[8] Richard White. What is Spatial History? Spatial History Lab: Working paper [online] `http://www.stanford.edu/group/spatialhistory/cgi-bin/site/pub.php?id=29` [Last accessed 2012-09-17]

[9] Kilian Schultes and Stefan Geißler. Orbis Latinus Online (OLO), Digital Humanities 2012, July 2012. `http://www.dh2012.uni-hamburg.de/conference/programme/abstracts/orbis-latinus-online-olo/` [Last accessed 2012-09-16]

Studies for Segmentation of Historical Texts: Sentences or Chunks?

Florian Petran

Department of Linguistics
Ruhr-Universität Bochum
E-mail: petran@linguistics.rub.de

Abstract

We present some experiments on text segmentation for German texts aimed at developing a method of segmenting historical texts. Since such texts have no (consistent) punctuation, we use a machine learning approach to label tokens with their relative positions in text segments using Conditional Random Fields. We compare the performance of this approach on the task of segmenting of text into sentences, clauses, and chunks, and find that the task gets easier, the smaller grained the target segments are.

1 Introduction[1]

Text segmentation is an important part of the natural language processing pipeline. Not only is it an important target for corpus annotation, since the sentence and constructions on the sentence level are subject to research, it is also an important pre-processing step for a lot of other annotations that presuppose a segmented text. Tasks such as bitext alignment or part-of-speech tagging either require a segmented text, or work better or faster with pre-segmentation.

Segmenting text into sentences is usually seen as a task of disambiguating punctuation marks, and there are several systems that perform that task well. But what if a text has no or no consistent punctuation to indicate sentence boundary? Syntactically motivated punctuation was not used until well in Early Modern times for European languages, or even as late as the 19th century for Chinese texts. Anyone working with historical texts will therefore be faced with the problem of having to manually annotate for sentence segmentation which is a non-trivial, time consuming task. Spoken language faces similar problems as well, but it has intonation features as indicators of sentence breaks, something which is largely absent in historical texts.

[1]The research reported here was financed by Deutsche Forschungsgesellschaft (DFG), Grant DI 1558/4-1.

There is already some related work on Chinese sentence segmentation, and spoken language segmentation (see section 2 below). Based on those results, this paper introduces experiments with segmentation of modern German language data using Conditional Random Fields (Lafferty et al. [5]) without the use of punctuation marks. They are aimed at developing a method of segmenting historical texts. In the results, we find that a chunking approach works far better than the sentence segmentation, which suggests that it would be prudent to do the sentence segmentation based on the chunks determined.

The paper is organized as follows. Section 2 introduces approaches this work is based on. The data used for the experiments, and the method used to obtain it is described in section 3. Section 4 describes the experiments, their results, and the method of evaluation. Finally, section 5 offers some concluding thoughts and directions for future research.

2 Related Work

Most sentence segmentation systems for Western languages rely on the presence of periods in the text. These need to be disambiguated between sentence terminating punctuation symbols, and punctuation in other functions such as abbreviations. In that, systems such as Gillick [2] achieve very good results with F-scores well over 90%. However, the solutions are not transferrable to a situation where punctuation is inconsistent, or entirely absent.

One such situation is spoken language data, where capitalization or punctuation do not exist at all. For such cases, a machine learning approach is used to classify tokens according to whether they constitute a sentence boundary or not (such as in Stevenson and Gaizauskas [10]). More recent approaches achieve good results with Hidden Markov Models or Conditional Random Fields (Liu et al. [6],[7]), but they use prosodic information such as timing, energy, or pitch as additional features. With the NIST SU detection data set, there is an established gold standard data set that is used by many researchers. No such data set exists for historical or modern German.

One of the key features of speech data is that it is not always straightforward to segment into sentences, since a sentence in spoken language is not as well defined as in written language: due to their spontaneous nature, spoken sentences may be interrupted, or syntactically incomplete. Furthermore, even in written language it is often subjective if a comma or a period should be used at a given point in the text. For research on spoken language data, this is approached with the definition of sentential units that are annotated in the transcribed data instead. An inter-annotator agreement of 93% for the NIST SU detection task (Liu et al. [6]) shows that even the manual annotation is not a trivial task. Spoken language data bears some similarity to historical texts in that the transcription is another potential source of error, whereas historical language has the normalization of spelling variants as error source. However, the key difference is that prosodic information,

used by all newer approaches to speech segmentation, is not available for historical texts.

Another important difference is that spoken language research focuses mostly on English, which has a relatively fixed word order. The results are thus not entirely transferrable to a language with more free word order, such as German. Nevertheless, an important result from Liu et al. [7] that influenced the experiments presented here is the improved performance of a CRF based system compared to HMM or MaxEnt based ones.

In Chinese, punctuation was not commonly used until well in the 19th century, and is often not used today in informal (Internet) communication (Huang et al. [3]). Huang et al. [4] employ a setup very similar to the one used in this paper, where CRF are used to label different data sources, and test different tagsets for sentence breaks. In addition to POS tags they use phonetic representation of the words indicating (among other things) the initial and final tone. It is worth noting that even though they achieve F-scores of up to 83% on historical texts, modern Chinese texts tend to have more complex sentences, and are therefore harder to segment and they achieve far lower results on them. Obviously, the segmentation task is easier, the simpler the units are.

3 Data Basis and Extraction Methods

So far, there are no complete historical treebanks of German that have suitable annotation for all tasks. Instead, the data used for the experiments is extracted from the Tübinger Baumbank des Deutschen/Zeitungstext (TübaDZ)[2], a treebank of about 65,000 sentences of German newspaper texts. Modern German uses capitalization to indicate nouns, but this is usually not the case for Early Modern German. Additionally, an important property related to the task is the absence of (consistent) punctuation. We therefore filtered the punctuation marks from the texts and downcased all tokens, so as to simulate a historical text as closely as possible. Another important property of historical texts is frequent occurence of variant spelling, which was not simulated. Instead, we assume a work environment where all historical tokens are already mapped to their modern equivalent spellings.

3.1 Data Source and Sentence Segmentation Task

For the annotation of boundaries, we used the 5-tags set that performed best for the Chinese segmentation task (Huang et al. [4]), described in Table 1.

As far as data extraction is concerned, the simplest possibility is to annotate the boundaries of a sentence (S node) in the data. For this annotation we need not take the parse tree into account and use no additional heuristics.

[2] http://www.sfs.uni-tuebingen.de/tuebadz.shtml

Tag	Meaning
L1	Beginning of segment
L2	Second token in segment
M	Middle of segment
R	End of segment
S	Singleton segment

Table 1: 5-tags set for sentence segmentation.

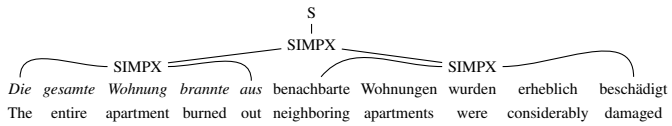


Figure 1: "The entire apartment burned out, neighboring apartments were severely damaged." — An example for embedded SIMPX with a straightforward solution.

3.2 Clause Extraction Task

Due to the recursive nature of clauses, their extraction from TübaDZ is not as straightforward. The TübaDZ parse tree is annotated with categories for simple clauses, relative clauses, and paratactic constructions of simple clauses, all of which are collectively referred to as SIMPX from here on. The extraction of these requires some heuristics, which are further described on some simplified example parse trees. First, a SIMPX node can itself contain other SIMPX nodes as the simplified parse tree in Fig. 1 shows. For these cases, we decide to always use the lowest SIMPX nodes, i.e. those further away from the root of the tree, and discard the higher level clause for segment annotation purposes.

Since German has a relatively free word order, this heuristics is not in all cases sufficient, as the example of a partial sentence in Fig. 2 shows. The simple clause begins at the first token *verantwortlich*, and then continues from *sei* until the end of the example. Beginning at the second token *so* up to *AWO*, however, is a paren-

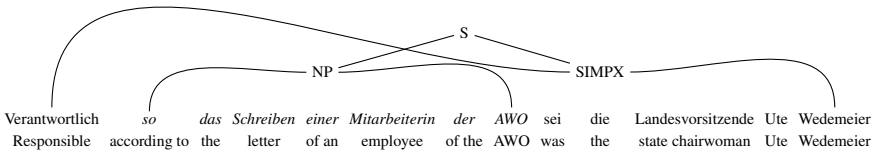


Figure 2: "According to the letter of an employee of the AWO, state chairwoman Ute Wedemeier was responsible."—An example of an embedded structure that is not a simple clause by itself.

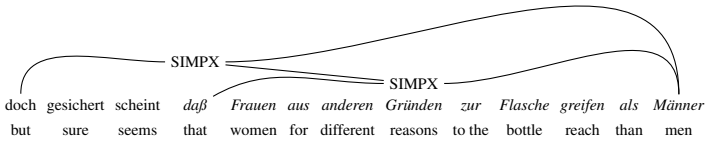


Figure 3: "but it seems safe to say that women take to the bottle for different reasons than men." — A SIMPX node that contains another SIMPX, but also terminals.

thetical NP that is independent of the simple clause, and therefore directly attached to the root sentence node S. In this case, the heuristics of taking the smallest node obviously does not suffice, since both nodes are siblings. So we have elements of the sentence that are not part of any clause from a syntactic point of view.

For these cases we would extract the NP in the example as a clause on its own. The way this was implemented was to gather all terminal symbols that were not extracted as part of a clause, and regard them as a segment as long as they were consecutive. Furthermore, the flat boundary annotation scheme is unable to account for non-consecutive tokens in a segmental unit. Since we cannot reorder the data, we extract the SIMPX in cases like these as two different segments, one starting at *sei* up to the end, and one with *verantwortlich* as its only word.

Fig. 3 shows another (partial) example sentence from the corpus. If we apply the rule to always use the smallest SIMPX in this case, we would be left with unassigned terminal symbols in the result, since the SIMPX starting at *daß* running up to the end is embedded into the superordinate SIMPX, but there are also terminal symbols. In order to not overcomplicate the heuristics, we applied the rules set above straightforwardly in this case as well. This means that we extract all terminals preceding the smaller SIMPX as a clause on their own, and the following SIMPX as an independent clause. Overall, this means that the clause annotations are less than perfect, since the nature of a parsed corpus makes it ill suited for a shallow segmentation task.

3.3 Chunk Segmentation Task

Finally, we extracted chunks to compare the chunk segmentation task with the sentence segmentation. Unfortunately, again, the TübaDZ annotation is not always well suited for a chunking task. Although there are node labels for verb phrases, these are in fact used to annotate the verbal head of the phrase instead of the phrase governed by the verb. We therefore excluded verb phrases from the extraction and added another tag O to the tagset, indicating that a token is outside of an annotated chunk. The chunks were extracted in a similar way to the SIMPX nodes, preferring bottom most nodes for the extraction, with the following additions to the heuristics.

If a prepositional phrase was present, it was preferred over a noun phrase, and if a noun phrase was present, it was preferred over an adjective phrase. The only other phrases that are annotated in TübaDZ are complex determiners and certain adverbial phrases, but since these are always embedded in a predictable way, we ignored them for the data extraction.

3.4 Data Properties

Table 2 shows the tag distributions for each task. Obviously, the number of L1 and R tags is always the same because a segment starting with an L1 tag has to be always closed by an R tag. The difference between the amount of tokens tagged L1 and L2 indicates the amount of segments that are longer than two tokens: a two token segment only has L1 and R tags, and only segments with three or more tokens have the L2 tag. The difference between the amount of L2 and M tagged tokens indicate the amount of segments that are longer than three tokens, because only those would have the M tag, but also how likely it is that these segments are exactly four tokens, or longer.

	L1	L2	M	R	S	O
Sentence	6.4954%	6.2716%	**80.5324%**	6.4954%	0.2052%	–
Clause	14.1560%	12.5402%	**57.3251%**	14.1560%	1.8226%	–
Chunk	19.1940%	11.1127%	18.8499%	19.1940%	9.1268%	**22.5482%**

Table 2: Relative frequencies of the boundary tags in % of 832,975 tokens. Most frequent tags for each task are bold face. The leftmost column indicates the task.

For the sentence segmentation task, the number of L1 and L2 tags is roughly the same, which indicates that there are only few two token segments—incidentally about the same number as one token segments (tagged S). The vast majority of tokens is tagged M, which indicates that most segments are well over four tokens long. For the clause segmentation task, the number of two token segments is, again, roughly the same as the number of one token segments, but there are a lot more of both categories. Compared to the sentence task, there is a lower proportion of M tags, again unsurprisingly, as the segments are relatively shorter, since they are subsegments of sentences. The chunk segmentation task has a much more even distribution of tags, with a small majority of tokens tagged O (for outside of chunk). The relative numbers of L1 and L2 tags indicate that a little less than half the segments are only two tokens long. But for those that are at least three tokens long, there is a high probability that they are either four tokens or longer.

4 Results and Evaluation

4.1 Evaluation Method

All the following results were obtained using 5-fold cross validation, averaging the results over all folds. The corpus was split along the sentence boundaries closest

to one fifth of the tokens, such as to minimize noise in the data. Training and application of the model was done using the CRF++ toolkit[3] with the default feature template, token and POS tag as feature, a minimum feature frequency of 2 and the default cost function of 1.0. All test and training data was converted to lower case.

In addition to testing with the gold standard POS data, we also annotated the test data with the TreeTagger (Schmid [9]) which ships with a parameter file for German. This was done to more closely simulate real world conditions, where text segmentation is hardly done on texts with manually corrected POS tags. Since the parameter file for the tagger expects mixed case input, we used the original, mixed case tokens for tagging and downcased them afterwards. However, explorative experiments suggest that tagging performance loss by retraining a German tagger on downcased data is negligible.

With a set of less than six tags, tagging accuracy has a fairly high baseline for the most frequent tag assumption. Evaluating tag accuracy as a percentage of correct tags is therefore not a very meaningful indicator of the overall performance of the system. Instead, we evaluated precision and recall for the segment boundaries in the text. This was determined as follows. If a token was tagged as the start of a segment, and if the immediately preceding token was tagged as the end of a segment, a boundary was assumed to be present between the tokens. Note that this means that the evaluation for the sentence tagset is theoretically slightly stricter than for the IOB tagset, since it relies on two tags being correct rather than just one. In practice, this does not matter so much, since the system reliably learned the correct sequence of boundary tags in all cases. A baseline for this evaluation approach is difficult to determine, and any random guessing baseline would be fairly low. For the purpose of this paper, however, it does not matter as much, since our focus is on the comparison of relative difficulties of the different tasks.

Where applicable, O–tagged tokens were counted in the same way as singleton phrases. Of course this means that a theoretical baseline is different for all tasks below (see section 3.4). The smaller the units that are tagged, the less M tagged tokens and the more S and O tagged ones appear. But none of the correct M tags count towards the performance reported below, since they are always negative categorizations.

4.2 Sentence Segmentation Task

The results for the sentence segmentation task are shown in Table 3. They are better than could be expected by random guessing, but certainly not good enough for real world application in this form. It seems that this task is especially sensitive to only slightly erroneous POS tags, so we performed an additional experiment with a simpler template using only the surface token as feature. If we compare those results to the results with generated tags, we notice, again, a marked decrease in overall performance, but not as much as from the gold POS tags to the generated

[3] http://crfpp.googlecode.com/

tags. In fact, the precision is even slightly higher without the tags.

	Precision	Recall	F-score
Gold POS Tags	71.57%	59.13%	64.76%
Generated POS	63.27%	48.72%	55.04%
Without POS	63.75%	40.15%	49.27%

Table 3: Results for the sentence extraction task.

The likely reason for the poor performance of this task is that sentence boundaries are, to some degree, subjective. It is often a matter of style if a writer uses a full stop, a comma, or a semicolon at a given point. This has a detrimental effect mainly on recall, as it is difficult to find all cases that had full stops in the original text, but it also effects the overall performance negatively, because many items in the training data are learned as in the middle of a sentence when they might as well have been the first token in a sentence. The comparatively higher precision suggests that a bootstrapping approach could possibly improve performance. The small effect of the generated POS tagging indicates that this might work even for a text where tagging proves difficult and could, in fact, be potentially used to improve tagging performance. But before such enhancements are considered, a more detailed error analysis would have to be done.

4.3 Clause Segmentation Task

For the simple clause boundary annotation task shown in Table 4, the performance is much better. This is especially remarkable given the tradeoffs from our clause extraction heuristics described in Sec. 3. Obviously, simple clause boundaries are easier to learn than sentence boundaries, even with noisy data. The noise also seems to affect recall slightly more than precision. In a real world application, such a task might benefit from annotation of sentential units instead of clauses or sentences, such as described in LDC [1].

	Precision	Recall	F-score
Gold POS Tags	83.00%	69.37%	75.57%
Generated POS	73.60%	58.67%	65.29%
Without POS	71.07%	54.44%	61.65%
Simplified data	88.25%	79.46%	83.72%

Table 4: Results for the clause extraction task.

The last line of Table 4 shows the results with gold tags for an idealized setting were only continuous SIMPX consisting of nothing but terminals were extracted. Note that this is not entirely comparable to the other rows, since the source data was adjusted, but it shows how performance might improve for an ideal language that did not force difficult decisions such as the ones described in Sec. 3 upon us. The effect this has on recall is roughly twice as high as the effect on precision. This

shows that even for the noisy data, the system learned the right clues reliably, but was less able to apply them in all cases.

4.4 Chunk Segmentation Task

For the chunking task, we did an additional run of the experiments with the IOB tagset commonly used for chunking (Ramshaw and Marcus [8]). The IOB tagset only knows annotations for a position in- or outside of a chunk (I and O respectively), or starting a new chunk (B). The experiments done by Huang et al. [4] suggest that the 5-tags set performs best, but they do not evaluate its performance on chunking, or the IOB tagset.

As Table 5 shows, the chunking task yields very good results for both tagsets. The 5-tags set task outperforms the IOB tagset by a small margin. As the quality of the POS annotation decreases, however, the gap between the two annotation schemes shrinks slightly. It is also noteworthy, that the chunking task is far less sensitive to the quality, or in fact the presence of POS annotation than the other tasks, for both annotation schemes, and even slightly less so for the sentence tagset.

	5 tagset			IOB tagset		
	Precision	Recall	F-score	Precision	Recall	F-score
Gold POS Tags	92.72%	94.90%	93.79%	91.82%	94.77%	93.27%
Generated POS	91.19%	90.78%	90.98%	90.37%	92.70%	91.52%
Without POS	90.79%	88.87%	89.82%	88.36%	89.07%	88.71%

Table 5: Results for the chunking task using the extended 5-tags set and the IOB tagset, and gold or generated POS data respectively.

Of the three tasks examined here, the chunking task is the only one where recall is higher than precision, if only by a small margin, but recall is also more affected by the POS annotation quality than precision. In any case, and for both annotation schemes, the chunking task yields good results even with non-optimized, standard settings for the learning program.

4.5 Error Analysis

Tables 6, 7, and 8 show the confusion matrices for the sentence, clause, and chunk segmentation tasks respectively. All matrices were generated for the results with gold POS tags. They show that all tasks perform far better a random guessing scenario, and for almost all tags and tasks, the correct guess is the vast majority. For the sentence and chunk segmentation tasks, all tags have a relatively high probability of confusion with the M tag, which is to be expected since it is by far the most frequent in those tasks. The low probability of confusion of the L1, L2, and R tags with each other show that where the system guessed the boundary wrong, it was not off by just one or two positions, but generally did not recognize that a boundary was present at all, tagging M instead. So far, this is exactly what should be expected.

		Expected				
		L1	L2	M	R	S
Actual	L1	**31514 (58.78%)**	535 (1.04%)	11972 (1.01%)	349 (0.65%)	378 (22.13%)
	L2	646 (1.20%)	**30114 (58.60%)**	11816 (1.79%)	867 (1.63%)	44 (2.58%)
	M	20872 (38.93%)	20058 (39.03%)	**624645 (94.62%)**	20420 (38.29%)	497 (29.10%)
	R	389 (0.73%)	674 (1.31%)	11650 (1.76%)	**31630 (59.32%)**	204 (11.94%)
	S	195 (0.36%)	9 (0.02%)	55 (0.01%)	58 (0.11%)	**585 (34.25%)**

Table 6: Confusion matrix for the sentence segmentation task. Cells indicate the number of guesses for the category denoted by the row, the percentage is of the row total. Correct results are bold faced.

		Expected				
		L1	L2	M	R	S
Actual	L1	**82300 (70.36%)**	2772 (2.69%)	12283 (2.62%)	1908 (1.64%)	2991 (20.05%)
	L2	4052 (3.46%)	**73995 (71.82%)**	118818 (2.52%)	3637 (3.12%)	461 (3.09%)
	M	27232 (23.28%)	23907 (23.21%)	**431551 (92.07%)**	27447 (23.56%)	3919 (26.27%)
	R	1768 (1.51%)	2217 (2.15%)	12256 (2.61%)	**82614 (70.90%)**	3036 (20.35%)
	S	1626 (1.39%)	134 (0.13%)	825 (0.18%)	915 (0.79%)	**4510 (30.23%)**

Table 7: Confusion matrix for the clause segmentation task.

Concerning the singleton segment S tag for the sentence task, if we disregard the confusion with M, the system fairly frequently confused it with L1. This indicates that it often correctly identified a sentence starting at that position, but incorrectly guessed the length of that sentence. Overall, the S tag has the lowest accuracy in that task, probably because it is very rare in the training corpus. The difference is even smaller for the clause segmentation task, where the correct guesses for S only barely outweigh the L1 guesses. This is in spite of the fact that the S tag for clauses is far more frequent than for sentences, though still comparatively rare. The likely source of that problem are the problems with data extraction for clauses described in Sec. 3.2. Since a lot of S tagged tokens are not actually singleton segments, but instead part of discontinuous clauses, it was difficult to learn them reliably. This is confirmed by the far better results for this task on the idealized data as described in the bottom row of Table 4.

		Expected					
		L1	L2	M	R	S	O
Actual	L1	**112679 (88.92%)**	1775 (2.48%)	10166 (8.32%)	227 (0.18%)	1190 (2.00%)	1233 (0.82%)
	L2	1359 (1.07%)	**60486 (84.35%)**	6676 (5.46%)	2576 (2.05%)	161 (0.27%)	328 (0.22%)
	M	8544 (6.74%)	5579 (7.78%)	**88104 (72.11%)**	7966 (6.32%)	1401 (2.35%)	2143 (1.43%)
	R	200 (0.16%)	2906 (4.05%)	10506 (8.60%	**109891 (87.25%)**	2145 (3.60%)	944 (0.63%)
	S	1343 (1.06%)	223 (0.31%)	2281 (1.87%)	2161 (1.72%)	**53946 (90.66%)**	67 (0.04%)
	O	2600 (2.05%)	742 (1.03%)	4440 (3.63%)	3131 (2.49%)	663 (1.11%)	**145220 (96.86%)**

Table 8: Confusion matrix for the chunk segmentation task.

The confusion matrix for the chunk segmentation task shows the lowest accuracy for the M tag, which is mildly surprising since it is not all that infrequent in the data. On the other hand, the tag frequencies are much more balanced overall for this task, so any effect of tag frequency would be smaller overall. One factor that is not accounted for at all by the results reported in Sec. 4.4 is the remarkably small confusion between O and S. Any such confusion would not be counted as word boundary error by our evaluation scheme described in Sec. 4.1, since they both serve the same function with regards to the word boundary. This indicates that a possible labeling of chunks with or after the segmentation could work well.

5 Conclusion

We have shown that it is generally possible to segment German text without the use of punctuation marks. As the target units for segmentation become smaller, this task becomes easier, to a point where it achieves results that would be suitable as basis for manual correction in a real world application. Further improvements might be achieved by optimizing the parameters for the CRF system. The reason for this performance increase is likely that sentences are, to a degree, subjective, and their segmentation dependent on individual style in many cases. Clause units have fairly complex relations with each other and are therefore not easily captured by a flat boundary annotation scheme.

Chunks on the other hand, as defined by the data extraction in Sec. 3, are simple constituents of sentences that are easily objectively identifiable. This suggests that instead of annotating sentences, it would be more efficient to first chunk the texts, and then use those chunk boundaries to identify sentential unit breaks. Since the chunks can never cross sentence boundaries, the sentence segmentation based on the chunk boundaries would be reduced to the disambiguation of chunk boundaries, similar to the way that written text is normally segmented. This is a far easier task than segmentation of text without any indications of boundaries since it limits the possibilities of where boundaries can be located.

One direction that needs to be explored for future research is the performance of the system on an actual historical text. As mentioned above, these types of text have large amounts of variant spellings that need to be normalized for a model trained on modern German to be applicable. Such a normalization step would introduce an additional source of error to both POS tagging and the segmentation, so it needs to be clarified how large the impact of an additional error source on the overall system will be. Alternatively, the segmentation model could be trained on non-normalized historical data. For this case, the effect on training corpus size would need to be explored.

As an intermediate step, it would also be helpful to determine the performance on a corpus that has annotations for verb chunks as well. We had to leave these out of the experiments entirely because the TübaDZ annotations do not allow an extraction of verb chunks. Had VP annotation been present, we would have less out of chunk elements, less word boundaries, and likely longer segments overall.

Furthermore, there are various improvements that could be made on the performance of the system itself. For example, Liu et al. [7] report that a majority vote between MaxEnt, HMM, and CRF classifiers performed better than each individual one for the task of spoken language segmentation. Such a majority vote system would be computationally expensive, but not very hard to implement. Another possibility is a bootstrapping approach, suggested by the fact that for most tasks, precision is far higher than recall.

Finally, another open question is the labeling of chunks, which would, ultimately, be a desirable outcome of the chunk segmentation. There are basically two different approaches to the shallow parsing task with regards to our segmentation

system. We could label the chunks after they have been segmented, or combine the chunk labeling with the segmentation for a larger tagset. Each approach could have its advantages, but discussing them is beyond the scope of this paper.

References

[1] Linguistic Data Consortium. Simple metadata annotation specification version 6.2. 2004.

[2] Dan Gillick. Sentence boundary detection and the problem with the US. In *Proceedings of NAACL*, pages 241–244. Association for Computational Linguistics, 2009.

[3] Hen-Hsen Huang and Hsin-Hsi Chen. Pause and stop labeling for chinese sentence boundary detection. In *Proceedings of RANLP*, pages 146–153, 2011.

[4] Hen-Hsen Huang, Chuen-Tsai Sun, and Hsin-Hsi Chen. Classical chinese sentence segmentation. In *Proceedings of CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 15–22, 2010.

[5] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

[6] Yang Liu, Nitesh V. Chawla, Mary P. Harper, Elizabeth Shriberg, and Andreas Stolcke. A study in machine learning from imbalanced data for sentence boundary detection in speech. *Computer Speech & Language*, 20(4):468–494, 2006.

[7] Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of ACL*, pages 451–458. Association for Computational Linguistics, 2005.

[8] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94. ACL, 1995.

[9] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12, pages 44–49, 1994.

[10] Mark Stevenson and Robert Gaizauskas. Experiments on sentence boundary detection. In *Proceedings of the sixth conference on Applied natural language processing*, pages 84–89. ACL, 2000.

Historical spelling normalization. A comparison of two statistical methods: TICCL and VARD2

Martin Reynaert, Iris Hendrickx and Rita Marquilhas

TiCC research group, Tilburg University, Tilburg, The Netherlands
Centro de Linguística da Universidade de Lisboa, Lisboa, Portugal

E-mail: reynaert@uvt.nl, iris@clul.ul.pt, rmarquilhas@fl.u.pt

Abstract

We present a comparison of two statistical tools for spelling normalization of historical Portuguese. The VARD2 tool has been originally developed for Early Modern English but has been successfully ported to the Portuguese language. The second tool TICCL was developed for English and Dutch. The VARD2 tool was explicitly developed for historical data, while TICCL aims to handle spelling and Optical Character Recognition variation in very large corpora of digitized 19th and 20th century text. Here we detail both tools, their methods, strengths and weaknesses and their performance on the task at hand.

1 Introduction

There are several reasons why spelling normalization of historical text is essential. Information extraction or retrieval tasks on a historical corpus cannot be handled by any standard search system. If the user would query for a particular modern key word in the corpus, such a system will not be able to retrieve all relevant matches for the query as the different spellings of a word will not be recognized and retrieved. Creating a corpus version that is normalized for spelling would alleviate this problem.

Furthermore, automatically adding linguistic information such as part-of-speech (POS) tags will be much easier on a normalized version of the historical text. Tools such as POS-taggers have been developed for contemporary text and these tools will make more errors when labeling unknown spelling variants [12]. [8] have shown that automatic normalization of the historical data leads to more accurate POS-labeling. In a previous study we have already shown that automatic normalization of the historical data leads to more accurate POS-labeling [8].

Another motivation for spelling normalization is that these old texts with many different spellings and archaic words are difficult to read for non-specialists. These

historical texts are a part of the country's cultural heritage that should be publicly available. A modernized version facilitates the accessibility of such historical texts for the general public.

Here we present a comparison between two different tools, VARD2 [2] and TICCL [16], for automatic spelling normalization of a corpus of historical Portuguese. VARD2 has already been compared against two other spelling checkers, the MS-Word spelling checker and Aspell on historical English text [13]. It was shown that VARD outperforms the other tools as it has a higher precision. TISC (Text-Induced Spelling Correction), the precursor of TICCL, has also been compared with Aspell, Ispell and the MS-Word spelling checker for contemporary Dutch and English text [15] and reached much higher precision than the other systems due to taking into account the full vocabulary of the texts to be corrected and due to making use of bigram information, thereby performing context-sensitive spelling correction on non-words. Bigram correction is now also implemented in TICCL.

In this investigation we apply both tools to Portuguese, a morphologically rich language substantially different from either English or Dutch, for which these tools were originally developed.

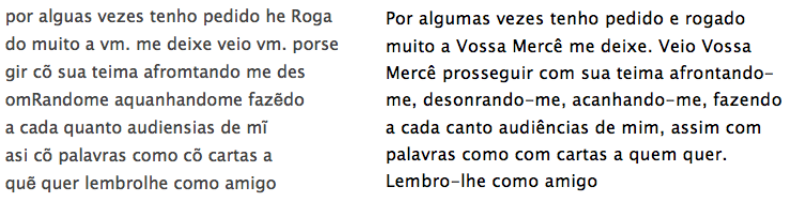
Moverover, the corpus used in these investigations is the Portuguese historical CARDS-FLY corpus that consists of digitally transcribed collections of personal letters and was developed for the historical study of Portuguese language and society. Obviously, handwritten personal letters contain more spelling variation than letters that were produced by professional scribes or clerks or than typeset and printed books that were published by printers.

In the next section we first give a brief account of the purpose and history of the CARDS-FLY corpus. In section 3 we discuss related work on spelling normalization for historical text. We present both tools in section 4 and present our experiments and results on a subset of the full corpus that was manually normalized for spelling in section 5. We discuss our findings in section 6.

2 Corpus

The study of language history can be supported by rather 'popular' sources that exist by the thousands, unpublished and ignored, on the stacks of public archives of the western countries. These sources are namely private letters, in our case, confiscated by courts of law. They differ from literary texts and from institutional documents in that they were not written for posterity or public reading. They were not written to be preserved, but they were nevertheless so. They were meant to circulate in the private sphere but, again, they went public. The judges of different courts - either religious, or civil or military - used the letters as instrumental proof, so their style quality was irrelevant, and they would do even if poorly written. The only thing that mattered was their referential contents.

In Early Modern Portugal, two different courts, namely the Inquisition and the



por alguas vezes tenho pedido he Roga
do muito a vm. me deixe veio vm. porse
gir cõ sua teima afromtando me des
omRandome aquanhandome fazẽdo
a cada quanto audiensias de mĩ
asi cõ palavras como cõ cartas a
quẽ quer lembrolhe como amigo

Por algumas vezes tenho pedido e rogado
muito a Vossa Mercê me deixe. Veio Vossa
Mercê prosseguir com sua teima afrontando-
me, desonrando-me, acanhando-me, fazendo
a cada canto audiências de mim, assim com
palavras como com cartas a quem quer.
Lembro-lhe como amigo

Figure 1: Example of a manually transcribed letter from 1592 addressed to merchandiser Joào Nunes. English translation: *I have more than once asked Your Honour and begged Your Honour to leave me alone. But Your Honour has insisted on defying me, dishonouring me, lessening me, engaging in gossip about me at every corner, both by words spoken and by letters written to whoever you choose. I remind you, speaking as a friend...*

Royal Appeal Court (Casa da Suplicação) collected and filed in the courts proceedings many of these letters. In the CARDS Project (Cartas Desconhecidas), 2.000 of such documents were detected, contextualised, and transcribed by a team of linguists and historians of the Early Modern period. The project ran from 2007 to 2010. The role of the linguists was to decipher and publish the manuscripts with philological care in order to preserve their relevance as sources for the history of language variation and change. The role of the historians was to contextualise the letters' discourse as social events. Almost half of the documents came from early 19th century criminal lawsuits of the Royal Appeal Court, and the other half from Inquisition lawsuits of the 17th and 18th centuries (plus a small sample from the 16th). In a complementary way (10 per cent), aristocratic families' legacies were also searched for private letters. The whole set of transcriptions, accompanied by a context summary, was given a machine-readable format, which allowed for the assemblage of an online Portuguese historical corpus of the Early Modern Ages.

As a sequel to CARDS, the FLY project (Forgotten Letters, Years 1900-1974) was launched in 2010 by the same core team, now accompanied by Modern history experts, as well as sociology experts. The aim was to enlarge the former corpus with data from the 20th century. Since collecting personal papers from contemporary times is a delicate task, given the need to guarantee the protection of private data from the public scrutiny, the letters of the FLY project come mostly from donations by families willing to contribute to the preservation of the Portuguese collective memory having to do with wars (World War I and the 1961-1974 colonial war), emigration, political prison and exile. These were also contexts favourable for a high production of written correspondence with family and friends because in such circumstances strong emotions such as fear, longing and loneliness are bound to arise.

The CARDS-FLY corpus [6] is thus a linguistic resource prepared for the historical study of Portuguese language and society. Its strength lies in the broad social representativeness, being entirely composed of documents whose texts belong to the letter genre, the private domain, and the informal linguistic register.

The current version of the CARDS-FLY corpus contains 3,455 letters with 1,155,206 tokens involving 2.237 different authors and addressees.

We show an example in Figure 2 of a digital transcription of a letter written in 1592, the version on the left side has the original spelling, apart from word boundaries normalization (except for enclitics), the right side was manually normalized for spelling[1]. This letter exemplifies the characteristics of this corpus of written historical letters: many letters do not contain punctuation marks, there are accents like the tilde that no longer have the same distribution in the current spelling, capitalization is used in a different way and does not signal sentence starts. Abbreviations such as *vm.* [P: Vossa Mercê E: Your Honour] are often used in these personal letters. Another difficulty is that there is not always a one-to-one mapping between words in the old and new spelling, since orthography was non-existing and creative spellings were far from rare, especially when writers were half-illiterate men and women.

3 Related work

As mentioned in the introduction, modernizing historical text aids information retrieval (IR) results. Another strategy is to adapt the search interface in such a way that it can cope with the spelling variation. This approach was taken by Gerlach et al. [4] who use a modern lexicon combined with transformation rules to expand the search query to capture also the spelling variants in the German historical text collection being searched. Other studies discuss several distance measures that augment the search query with fuzzy string matching [9] or acquire edit distance weights using unsupervised learning techniques [7].

A tool that was specifically developed to normalize the spelling of full texts, is the VARiant Detection (VARD) tool [13] that was developed for Early Modern English. We will discuss in detail its follow-up, the tool VARD2, in section 4.1.

Craig and Whipp [3] developed a method for automatic spelling normalization for early modern English. They combine list lookup of variants together with more sophisticated methods based on approaches taken in Word Sense disambiguation tasks to resolve ambiguous spelling variants that can be normalized to multiple modern forms.

The Historical Dictionary of Brazilian Portuguese (HDBP) is constructed on the basis of a historical Portuguese corpus of approximately 5 million tokens. As there was no standard spelling at the time (16th to 19th century), it is not easy to create lexicographic entries on the basis of the corpus or to produce reliable frequency counts. Therefore [5] developed an automatic rule-based variant detection method and created a spelling variants dictionary containing approximately 30K clusters of variants (we refer to this list as the HDBP-variant list).

[1] Full description at: `http://alfclul.clul.ul.pt/cards-fly/index.php?page=infoLetter&carta=CARDS4006.xml`

The Corpus of Early English Correspondence (CEEC) [11] is a corpus similar to the CARDS-FLY corpus as both corpora consist of two collections of historical letters, although the CEEC, contrary to CARDS-FLY, is not based on previously unpublished material. A start is said to be made with a spelling normalization step using the VARD2 tool.

4 Tools

4.1 VARD2

The VARD2 tool [2] is a Java program with several options for the normalization of spelling variation in text. The tool offers an interactive mode in which the program suggests a list of candidates for each unknown word in the text and allows users to select the best choice in a manner similar to the Microsoft Word spelling correction module. The tool can also be used to automatically correct a full text and it can be trained and tuned by the user for a specific data set. In these experiments we opted for this last option.

VARD2 works as follows. Each word is checked against a modern lexicon. Words that are not present in the lexicon are potential spelling variants. Note that this limits VARD2's capacity to the detection of non-word errors. For each potential spelling variant, a list of candidate modern counterparts is generated using a variant list consisting of pairs of variants and their modern counterparts, a character rewrite rule list and a Soundex algorithm to find phonetically similar counterparts. These modules together determine the confidence weight that is assigned to each candidate modern equivalent. VARD2 has a confidence threshold that determines what weight is needed to actually replace the variant with the highest weighted modern equivalent that exceeds the minimum threshold. If no likely candidates are found, the variant is kept.

Hendrickx and Marquilhas previously adapted the VARD2 tool for the Portuguese language [8] and here we use their Portuguese version of the tool. They replaced the English versions of the modern lexicon, the variant list with pairs of variants and their modern counterparts and the rewrite rule list with Portuguese versions. As variant list they used the HDBP-variant list combined with a variant list extracted from training material. The rewrite rule list is based on the edit rules automatically generated by the DICER tool [1] which takes as input a list with spelling variants and modern equivalents and extracts character string transformation rules to capture the spelling variation. Only those rules that occurred 5 times or more were retained. The rules in this set that were too generic were manually edited to be made more specific. The final rewrite rule set of VARD2 consisted of 99 rules. VARD2 can be trained on a data set that is already manually normalized. When VARD2 is being trained, the program adds all normal words to the modern lexicon and adds all variants from the training material and their frequencies to the variant list. The different confidence weights for each replacement method are also adapted on the basis of the training data.

4.2 TICCL

Text-Induced Corpus Clean-up (TICCL) is described more fully in [16]. It is now a fully-fledged web application and service due to CLARIN-NL project TICCLops. The main lexical variant look-up mechanism in TICCL is based on anagram hashing. Informally, this works as follows: for all the words in the lexicon and in the corpus, a numerical representation called the anagram value is calculated by making the sum of the code page values of the individual characters of the word raised to power five. The numerical value obtained is used as the index key in a hash, the actual symbolic word(s) having this value are added as the hash value. Words consisting of the same bag of characters will have the same anagram value. This is why this is called anagram hashing. Given the anagram for a particular word (or set of anagrams), called the focus word, TICCL builds list A which contains the anagram values for all the individual characters in the bag of characters as well as the anagram values for all the possible combinations of any two characters in the bag of characters. TICCL also has list B, which has the same for all the characters in the alphabet. Given a focus word, all its near neighbours can now be found by exhaustively subtracting any value from list A from the focus word's anagram value and adding any value from list B. If the resulting anagram value is present in the anagram hash, a numerical near neighbour has been detected. Retrieving the symbolic values, i.e. the actual word or set of anagrams, the edit distance to the focus word for each needs to be checked. Correction Candidates or CCs are those instances that differ by less or equal the number of edits allowed by the Levenshtein distance (LD), the limit of which is set by a TICCL parameter.

The number of look-ups required per focus word depends on the size of the alphabet. In prior work, in order to reduce the search space, TICCL was used with a reduced alphabet. Its lexicons and the corpus it works on were thoroughly normalized by e.g. rewriting any character bearing a diacritical mark as a single digit and all punctuation marks as another digit, all numbers and digits present having been normalized into as single, different, digit. In this work, we retain all unicode points below a high unicode number. This has the benefit of not having to restore or retrieve the original diacritical word form for output purposes.

Further, we have worked only in what [16] calls **focus word mode**, the corpus not being very large. In character confusion mode, TICCL scales to the largest corpora.

New in the present work is that TICCL has been applied to Portuguese and has been equipped with both absolute correction [10] and bigram correction capabilities.

Converting TICCL to Portuguese involved little more than providing it with a Portuguese lexicon, which was the same one as used for VARD2. Derived from the lexicon is a **word confusion matrix** by applying TICCL's character confusion module. In its essence this matrix is a list of the anagram value references between each word in the lexicon and all the other words in the lexicon that are reachable within the confines of the particular Levenshtein or edit distance set. In the present

work we have limited ourselves to LD set to 2 edits.

This word confusion matrix in fact provides the list of all possible **confusables** (also known as real-word errors in spelling correction or 'false friends' [14]). The operative definition of confusables is therefore that they are those words that can be formed from any given focus word in the lexicon by applying at most the number of edits implied by the LD handled. Use of the word confusion matrix allows for preventing the system from returning a valid lexicon word for any given valid (because present in the lexicon) focus word. This in fact implies that in its current implementation, TICCL cannot perform real-word correction. We will return to this matter in the discussion of the results.

Also in the current implementation, **bigram correction** is applied. In terms of the informal discussion of how TICCL works above it is easy to see that given the anagram values for all combinations of two consecutive words in the corpus, the corpus bigrams, the same mechanism can be applied to retrieve bigram CCs. This is because the numerical distance between the anagram values for e.g. the English words 'cat' and 'rat' will be the same for the likely bigrams 'the cat' and 'the rat' or 'white cat' and 'white rat', i.e. the numerical anagram value distance for 'c' to 'r'. Bigram correction is here applied only to short words. In prior work, a lower word length threshold was always applied. The word length threshold in our unigram mode experiments here was set to six characters. For very short words the lexical neighbourhood is very dense, substituting just one or two characters leads to very many other short words. We here try to overcome this problem for short words by looking in the corpus for variants for the bigrams they occur in. In doing so, TICCL handles short word bigrams as if they were just ordinary unigrams, the only difference being that now the space character is also at play. In its essence, by searching in only the bigrams containing a particular short focus word, the possible search space is effectively and efficiently reduced by the contexts it shares with potential near neighbours. Having retrieved variant bigrams, the overlapping, exactly matching left- or right bigram part is then discarded and the remaining pair of unigram variants is further handled in exactly the same way as the longer word pair variants which would have been retrieved. This approach has its limitations and this too will be further dealt with in the results section.

The **absolute correction** strategy was defined by [10], who called it 'limited but very cost-effective'. We equipped TICCL with absolute correction capabilities based on the collection of lexical variants as present in the training set. Put in a misspelling dictionary, when one of the known historical variants is encountered, it is simply replaced by the contemporary form.

5 Experiments

For the evaluation experiments of TICCL and VARD2 we use a subpart of 200 letters from the CARDS-FLY corpus. These letters were manually normalized by one linguist but difficult cases were discussed with a second expert. This data set

was split in 100 letters for training and tuning the tools, and 100 letters were set apart as a true test set. The test set contains 37,372 tokens of which 6,978 (19%) are spelling variants that need to be detected and normalized by the tools. We measure the performance of the tools and compute accuracy, recall, precision and their harmonic mean, F-score, on the spelling variants.

In our experiments with TICCL on the training set we learned that absolute correction using all the pairs in the variant list was highly detrimental to precision. In the end we settled on a subdivision where words that are only in the corpus and not in the lexicon were allowed to be absolutely corrected. Words that are in both were evaluated on whether they were ambiguous or not, in the sense that they have more than a single possible resolution in the variant list. Those that were ambiguous were not let to be absolutely corrected, with the one exception of the pair 'q-que'; the others were let be handled by TICCL's proper correction mechanism. The ones that were not ambiguous were not corrected, with the exception of three pairs ('hum-um', 'porem-porém'[2] and 'exmo-excelentíssimo'), which were absolutely corrected. Absolute correction, when applied, was given precedence over whatever TICCL had retrieved in all cases. For ambiguous cases we retained the most frequent variant in the variant list only, which results in a single CC for all instances of absolute correction, enhancing ranking.

VARD2 was trained on the training material of 100 letters and this tuned each of the modules for this particular data set. We used VARD2 in its "batch" mode in which each detected spelling variant is automatically replaced by its best-first ranked word form.

In table 2 we show the best-first ranked performance of TICCL and VARD2 on the test set. The results reported for TICCL are those obtained when TICCL had access only to the variant list obtained from the training set. TICCL2 reports results obtained when TICCL had access both to the list obtained from the training set, as well as to the HDBP-variant list. VARD2 was trained on both.

Tool	acc	prec	recall	f-score
VARD2	94.65	96.99	73.63	83.71
TICCL	93.25	94.27	67.96	78.98
TICCL2	93.50	94.38	69.33	79.94

Figure 2: Best-first ranked results on the test set of 100 letters

5.1 Test results analysis

It should be noted that the results reported are necessarily precision and recall scores on tokens, not on word types because of the ambiguity of part of the original tokens which may have to be resolved to different contemporary word forms.

[2] Actually, 'porem' is ambiguous in Modern Portuguese but not in this corpus.

Tool	acc	precision	recall	f-score
TICCL-bi-rank3	94.11	94.62	72.57	82.14
TICCL-bi-rank5	94.35	94.71	73.89	83.01
TICCL-bi-rank10	94.55	94.78	74.92	83.69
TICCL-bi-rank20	94.66	94.82	75.52	84.08
TICCL-uni-rank20	94.42	95.03	73.99	83.20
VARD2-notraining	90.58	93.79	53.05	67.77
TICCL-bi-rank20-noabsolut	89.18	92.03	46.02	61.35

Figure 3: Results on the test set of 100 letters measuring TICCL's 3, 5, 10 and 20 first-best ranking with bigram correction and with absolute correction. Also shown is the effect of TICCL not performing bigram correction. Finally, the effects of VARD2 and TICCL not having been trained/using absolute correction with the variation list(s)

The accuracy of the original corpus before correction is 81.33%. This means that less than 20% of the original texts need to be normalized. VARD2 manages to improve texts by 13.32%. TICCL manages an improvement of 11.52% and TICCL2 reaches 12.17%.

VARD2 returns only best-first ranked results. These are compared with TICCL's best-first ranked results in Table 2. VARD2, being specially trained on manually edited rules specific for the task, is the clear winner. TICCL has not received any special training, but has had bigram correction at its disposal and has been equipped with new code for dealing with the absolute correction.

Results reported in the upper half of Table 3 show clearly that if TICCL's ranking mechanism might be improved, it can potentially best VARD2. These are results obtained when TICCL's absolute correction had access only to the variant list obtained from the training set. Also, in these experiments, TICCL lacked the benefit of a background corpus of contemporary Portuguese bigrams.

In the lower half of Table 3 we show the results of a few ablation tests. First we give the result of running TICCL in unigram mode only, then we show what VARD2 and TICCL manage to accomplish without having the benefit of the information in the variant list(s).

In unigram mode only TICCL is in fact a bit more precise. But it necessarily loses recall: it has not itself retrieved any variants for words shorter than six characters other than the ones it has been able to resolve through the absolute correction. This clearly shows the improvement due to the bigram correction.

Further in the same table we also show performance results obtained when the systems have not had the benefit of the domain specific variant list(s) either for training or for the purpose of absolute correction. Both benefit a great deal, but TICCL clearly most. As the CARDS-FLY corpus consists of data from a very specific textual genre with typical characteristics, we observe that training and tuning

the spelling checkers explicitly on this genre, leads to a substantial performance gain.

6 Discussion

The results of the comparison of TICCL and VARD2 shed valuable light on the problem of historical spelling normalization.

First and foremost, the results show that there is an upper limit to what can be achieved with what is essentially non-context sensitive correction on historical data. Probably neither of the systems in these tests have reached this upper bound, but both nevertheless get close to it.

The situation is that TICCL's absolute correction and VARD2's equivalent, a special purpose rule, work splendidly for cases such as 'q' which unambiguously should be normalized to 'que'. However, the instances of the single character 'v' in the historical letters are variously resolvable to 'via ' (on just 1 occasion), 'vossa' (42 times), 'vossas' (17 times) and 'vosso' (just once). The corpus contains many more similar instances. This implies full-fledged context-sensitive correction, which neither system can currently provide.

It would be legitimate to wonder if the task undertaken here should not be divided over two separate tasks, to be handled possibly by different systems and evaluated separately. The results show that about one quarter of the test instances cannot be solved by either of the systems. This implies that either the systems need to be equipped with mechanisms that do allow them to be solved, or that other systems or approaches should be sought and applied.

As it is, the systems are measured in part on test instances they were not designed to be able to handle. This in part obscures their capabilities of handling what they were meant to be able to handle.

TICCL will probably never be set to handle spelling variation exceeding LD 4. Even applying LD 3 would have an adverse effect on its precision. It could nevertheless, much in the way VARD2 is, be taught on the basis of the variant list(s) to look for specific higher edit distance variants. The variant list has, e.g. the pair 'exmo', an abbreviation, which should be expanded to 'excelentíssimo'. This is currently handled by the absolute correction, but the test set might as well have the pairs 'exma-excelentíssima', as well as the plural forms. By teaching TICCL to look for the anagram value for 'elentíssi', this desirable generalization would be achieved. This we will implement in future work.

The results in Table 3 over the various ranks show that TICCL potentially reaches the same or an even higher level of performance as VARD2 currently does. Our conversion of TICCL to Portuguese so far has been inadequate in that it cannot deal with the higher degree of morphological variation in Portuguese compared to Dutch and English which it had so far been applied to. Also, in these experiments it turned out that the ranking mechanism described in [16] did not deliver the results hoped for. The performance results reported in Table 3 were obtained by

leaving out the frequency information from the final ranking of CCs retrieved. In the absence of a large, contemporary background corpus, the frequencies observed in the historical test corpus were too sparse or totally unavailable for contemporary word forms and their bigram combinations. Best results, as reported, in these tests were obtained by the combination of ranking on frequency of character confusion observed and LD, only.

A fruitful path for future work is to study the strengths of both systems and to see how these might be combined. The DICER tool provides a wealth of statistics on the variation present in the training set. Certainly TICCL would benefit from direct use of these statistics in its ranking of CCs. Also, we should study how to address more properly the problem of morphological variability in TICCL's ranking. In conclusion, some generalization from the information available to TICCL in the absolute correction list should be of benefit. Certainly for higher LD variation which is highly present in these historical texts due to the high incidence of abbreviations, providing TICCL with the common character confusions above the LD limit it is set to work with, would allow it to emulate VARD2 in that it would then also be trained to explicitly identify and retrieve these variants.

7 Acknowledgements

TICCL has further been developed by Martin Reynaert in the Dutch NWO project Political Mashup. Iris Hendrickx was funded by FCT Doctoral program Ciência 2008. Rita Marquilhas is supported by the projects FLY (PTDC/CLE-LIN/098393/ 2008) and Post Scriptum (ERC, Adv Grant 2011, GA 295562).

References

[1] A. Baron. *Dealing with spelling variation in Early Modern English texts*. PhD thesis, University of Lancaster, Lancaster, UK, 2011.

[2] A. Baron and P. Rayson. VARD2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*, 2008.

[3] H. Craig and R. Whipp. Old spellings, new methods: automated procedures for indeterminate linguistic data. *Literary and Linguistic Computing*, 25(1):37–52, April 2010.

[4] A. Ernst-Gerlach and N. Fuhr. Retrieval in text collections with historic spelling using linguistic and spelling variants. In *Proceedings of the ACM/IEEE-CS Conference on Digital Libraries*, pages 333–341, 2007.

[5] R. Giusti, A. Candido, M. Muniz, L. Cucatto, and S. Aluísio. Automatic detection of spelling variation in historical corpus: An application to build

a Brazilian Portuguese spelling variants dictionary. In *Proceedings of the Corpus Linguistics Conference*, 2007.

[6] M. Gomes, A. Guilherme, L. Tavares, and R. Marquilhas. Projects CARDS and FLY: two multidisciplinary projects within linguistics. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[7] A. Hauser and K. Schulz. Unsupervised learning of edit distance weights for retrieving historical spelling variations. In *Proceedings of the First Workshop on Finite-State Techniques and Approximate Search*, pages 1–6, Borovets, Bulgaria, 2007.

[8] I. Hendrickx and R. Marquilhas. From Old Texts to Modern Spellings: An Experiment in Automatic Normalisation. *Journal for Language Technology and Computational Linguistics (JLCL)*, 26(2):65–76, 2011.

[9] S. Kempken, W. Luther, and T. Pilz. Comparison of distance measures for historical spelling variants. In *Artificial Intelligence in Theory and Practice*, volume 217, pages 295–304. Springer Boston, 2006.

[10] J.J. Pollock and A. Zamora. Automatic spelling correction in scientific and scholarly text. *Commun. ACM*, 27(4):358–368, 1984.

[11] H. Raumolin-Brunberg and T. Nevalainen. Historical sociolinguistics: The corpus of Early English Correspondence. *Creating and Digitizing Language Corpora*, 2: Diachronic Databases:148–171, 2007.

[12] P. Rayson, D. Archer, A. Baron, J. Culpeper, and N. Smith. Tagging the Bard: Evaluating the Accuracy of a Modern POS Tagger on Early Modern English Corpora. In *Proceedings of the Corpus Linguistics Conference (CL2007)*, University of Birmingham, UK, 2007.

[13] P. Rayson, D. Archer, and N. Smith. VARD versus Word: A comparison of the UCREL variant detector and modern spell checkers on English historical corpora. In *Proceedings from the Corpus Linguistics Conference Series*, volume 1, Birmingham (UK), 2005.

[14] U. Reffle, A. Gotscharek, C. Ringlstetter, and K. Schulz. Successfully detecting and correcting false friends using channel profiles. *IJDAR*, 12(3):165–174, 2009.

[15] M. Reynaert. *Text-Induced Spelling Correction*. PhD thesis, Tilburg Universtity, 2005.

[16] M. Reynaert. Character confusion versus focus word-based correction of spelling and OCR variants in corpora. *International Journal on Document Analysis and Recognition*, 14:173–187, 2010. 10.1007/s10032-010-0133-5.