Ruef, Beni; Ueberwasser, Simone (2013): The Taming of a Dialect: Interlinear Glossing of Swiss German Text Messages. In: Zampieri, Marcos; Diwersy, Sascha (Hrsg.): Non-standard Data Sources in Corpus-based Research. (=ZSM-Studien, Schriften des Zentrums Sprachenvielfalt und Mehrsprachigkeit der Universität zu Köln 5. Hrsg: Bongratz, Christiane M.; Riehl, Claudia M.). Aachen: Shaker, 61-68.

The Taming of a Dialect: Interlinear Glossing of Swiss German Text Messages

Beni Ruef[1], **Simone Ueberwasser**[2]
Romanisches Seminar[1],
Deutsches Seminar[2]
University of Zurich

Abstract

The Swiss German dialects (SGD) have no fixed orthography. Furthermore, text messages contain a lot of abbreviations and forms of code-switching. To enable corpus search and the application of computational linguistic methods, the Swiss SMS corpus was normalized by means of interlinear glossing. This paper describes the tool developed for this task and the practical experiences gained.

1 Working with dialectal data

Until a few years ago, the written use of SGD[1] was limited to local poetry and literature. There has never been a standardized spelling system for this variety, not least because the individual dialects in the different regions are very heterogeneous. Since the rise of the new media, however, SGD has found its way into written communication, albeit for conversations of an informal nature. In spite of this frequent use of the dialect, there are

[1]More information about the SGD and the need for standardization can be found in Ueberwasser in this volume.

still no spelling norms, a fact causing major difficulties when creating or working with linguistic corpora. The written SGD is in fact a beast to be tamed. In this paper, we will show some steps taken by the Swiss SMS team (cf. www.sms4science.ch as well as [3]). towards this aim. In the second section, we will give an introduction to glossing and elucidate why we developed our own tool for the task at hand. Sections three and four will introduce the functionality and technology, respectively. In the last two sections, we will share our experiences and conclusions.

2 The need for a tool of our own

The Swiss SMS team was not the first one with a need for interlinear glossing. In fact, the approach of a sub- or superscripting word by word translation—thus the term *interlinear glossing*—can already be found in medieval texts, where individual Latin tokens were translated into the vernacular in this way. This glossing does not result in syntactically correct texts in the vernacular and thus cannot be used for all types of linguistic studies. However, it is still a valuable mean to trace individual word forms, to aggregate spelling variants etc. We invert this procedure, i.e. we take non-standardized spelling forms and apply an interlinear glossing in a standardized variety. Since this is a well-known approach to non-standard language, various freely available software tools can be found. In the following, we will briefly explain why two of the best known of them, ITE (cf. michel.jacobson.free.fr/ITE/) and VARD2 (cf. www.comp.lancs.ac.uk/~barona/vard2 and [1]), did not fulfill our specific requirements.

These requirements can mainly be traced back to the sheer amount of more than 10,000 SMS which had to be annotated, and to the complexity of the data. The latter resulted in the need for a user interface which allowed to see both the original and the glossed level at the same time so as to compare the two versions of the complete SMS (not possible in VARD2). Furthermore, the option of adding a gloss only was not sufficient since metadata had to be added, too (not possible in VARD2 and tedious in ITE), as will be shown in section 3. The amount of data, on the other hand, asked for a workflow which allowed for fast processing, such as by

suggesting glosses or by allowing the original token to be copied into the gloss with one click (not possible neither in VARD2 nor in ITE).

With some restrictions, the two tools mentioned above could both have been configured to fulfill some of these needs in one way or another. However, in the project we had the need to divide the work on several people to have the job done faster, a requirement which asks for a server-based solution. Neither of the existing tools offers this option. The focus here is not on being able to work on the same messages at the same time but on writing the glosses and their dialectal counterparts back to a common vocabulary list. This list is then used to produce suggestions for glosses to the whole team, a procedure which both helps in cases of doubt and speeds up the workflow. The next section will explain the functionality of this approach.

3 Functionality

The user interface of the SMS Glossing Tool (*SGT*) is comprised of several windows which reflect the three main tasks at hand: 1) the *overview panel* displaying the messages to gloss and their status[2], 2) the *detail panel* showing the selected SMS, and 3) the *glossing panel* with the tokenized SMS to be glossed (cf. Figure 1).

In the detail panel the editor can manually change the SMS' status to one needing attention (*needs retokenizing*, *marked for inspection*) and add a note describing the problem at hand. To speed up the workflow, messages which are completely glossed and contain no note will change their status automatically to *completed* when the next SMS is selected.

The main work is done in the glossing panel which is arranged in five rows and as many columns as the SMS contains tokens. The first row (*Message*) contains the original SGD SMS and the second one (*Gloss*) the interlinear gloss in Standard German. The latter starts empty except for the tokens which are copied as is (punctuation, emoticons, numbers, and spotted names as identified by prior tokenization and anonymization of

[2]The SMS were glossed in batches containing 250 messages each.

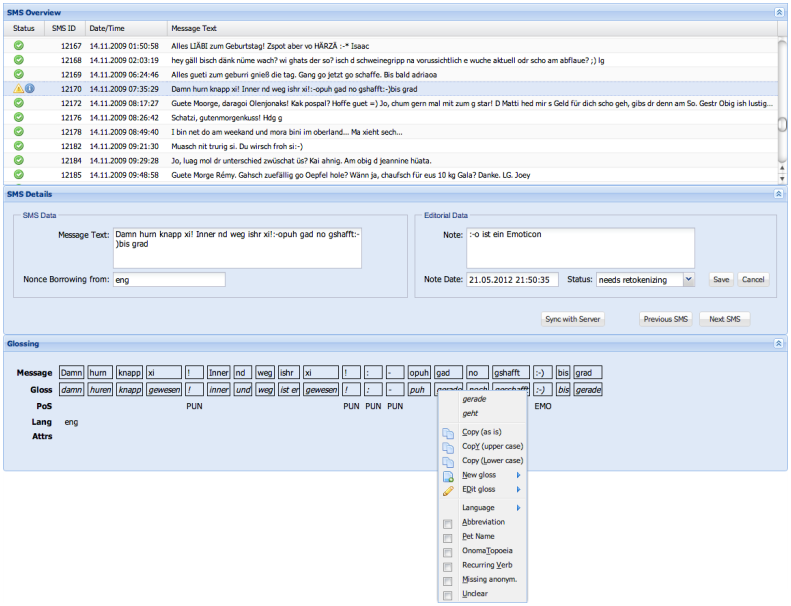

Figure 1: SGT user interface

names). The third row (*PoS*) indicates the part of speech, likewise identified by tokenization and anonymization of names. Code-switching on a token level is annotated in the fourth row (*Lang*), which also starts empty. The fifth row (*Attrs*) shows the attributes of a token which we consider interesting for further linguistic examination (cf. below); besides *abbreviation* (recognized by the tokenizer) this row also starts empty.

Clicking into a gloss' cell opens a menu (cf. Figure 1) which was designed with efficiency in mind. On top a gloss can be selected from a list of suggestions drawn from the vocabulary. In many cases the gloss is homograph to the SGD token and can be copied using one of the next three menu items. If the gloss to be added is neither in the list nor homograph, a

new gloss can be added. Erroneous glosses can be corrected by means of the next menu item. The remaining menu items allow adding metadata to a token. Firstly, the language of a nonce borrowing can be selected from a submenu. Additionally, the following boolean attributes can be added by the checkboxes below: *abbreviation*, *pet name*, *onomatopoeia*, *recurring verb*[3], *missing anonymization*, and *unclear*.

The vocabulary (i.e. the list of SGD tokens and their possible glosses) starts empty and grows progressively with each new gloss added. Any modification of the vocabulary is synchronized with the server, thus making new glosses available to the other editors immediately. The ever-expanding vocabulary fulfills two functions at the same time: it speeds up the process and it helps to increase consistency.

4 Architecture and technology

As outlined in section 2 we chose a server-based solution, or more precisely, a client-server architecture. Furthermore, we went for a web-based solution to manage without any software installation on the client's part. Virtually all application logic—after having been loaded from the server—is executed on the client side, the server—besides providing persistent data storage—is only in charge of vocabulary synchronization (cf. above).

The client side is implemented in JavaScript, using the ExtJS framework[4] whereas the server part is realized in Perl. The SMS data is stored in two XML files, one for the original messages and their metadata (status, notes etc.), and one for the tokenized messages, their glosses, and the metadata on the token level. The vocabulary is stored as a JSON [2] object where the SGD tokens themselves are the keys to arrays of possible glosses. The JSON format results in zero storage overhead and thus minimal download time when synchronizing the vocabulary while simultaneously allowing for extremely fast lookups.

[3] A reduplication of specific verbs, (mostly *gehen* 'to go', *kommen* 'to come', and *lassen* 'to let'), which is often compulsory in the SGD but unknown in the standard language.

[4] http://www.sencha.com/products/extjs

When the editor starts a new browser session the two XML data files are fetched from the server and used to initialize two corresponding DOM objects. All manipulation of the SMS and their tokens (adding a note, adding a gloss, setting an attribute etc.) results in updating one or both of the DOM objects. When the *Sync with Server* button is hit the two DOM objects—if altered at all—are serialized to XML again and saved on the server by means of an (asynchronous) XMLHttpRequest (XHR) using the PUT method. The vocabulary is synchronized whenever the editor switches from one SMS to another.

For post-processing (cf. section 5) the XML data files were stored in an XML database (BaseX[5]) where they were queried and modified by XPath 2.0 [4] and XQuery Update [5], respectively.

5 Post-processing

The more than 10,000 messages were glossed over a period of seven months. During this time the guidelines and specific rules for normalization were constantly improved based on the linguistic data encountered and the know-how accumulated so far. In addition, the tokenizer—which is crucial for the task of interlinear glossing—was continuously enhanced, e.g. because of emoticons not known in advance. Both phenomena cause a typical 'moving target' situation where all the changes must be recorded such that the resulting inconsistencies can be taken care of afterwards.

Tokenization errors following a regular pattern were corrected with XQuery Update. In the case of emoticons this can be done automatically. Some abbreviations[6] however need manual intervention as shall be demonstrated with an example: The two tokens *Fr* (or *fr*) followed by ‹.› always stand for an abbreviation, i.e. they must be retokenized to one single token *Fr.* (or *fr.*) and the abbreviation attribute must be set to true. The corresponding gloss, though, is ambiguous: *Fr.* can stand for either *Franken*, *Frau*, or *Freitag*.

[5] http://basex.org

[6] Most abbreviations used in the corpus were not known from the beginning and had to be retokenized in the post-processing step.

The correction of glossing inconsistencies caused by changing normalization rules is a similar case: Inconsistencies can quickly be pinpointed by means of XPath or XQuery queries, but their correction often involves manual work, too, mostly because of inflection, i.e. diverse inflected forms.

The systematic examination of all messages having the status *needs retokenizing* or *marked for inspection* not only revealed tokenization errors but also helped to detect other problems and inconsistencies as well, especially erroneous language taggings on the SMS level[7]. In a parallel quality assurance effort all tokens with the attribute *missing anonymization* were scrutinized, resulting in a large improvement of the corpus' anonymization. All the manual work listed above, be it control or correction, caused an enormous workload after the initial glossing process. Five editors were engaged in the glossing, spending an average 3.5 minutes per SMS[8]. Manual control and correction took up just as much time. The original editors working in the project had support from three different sides: a) the aforementioned vocabulary which suggested glosses, b) an extensive documentation which described the procedure in general linguistic terms, contained specific rules and also quoted many individual word forms, and c) an online forum in which the editors discussed problems among themselves. In spite of these tools, problems were frequent but will be shown on only one example in the following. The instruction was to stick as closely as possible to the dialectal form of a token and to consider any form as valid which can be found in at least one dictionary, even if marked as *colloquial* or *substandard* there. Such a tolerant stand towards the Swiss Standard variety clashes strongly with the teaching at basic schools where the standard variety, as it is used in Germany, is still seen as superior to the Swiss one. Thus, despite the instructions, the editors nevertheless often followed their instincts instead, resulting e.g. in glossing the verb *to move* as the German form *umziehen* instead of the Swiss form *zügeln*, notwithstanding the latter being registered in different dictionaries. The vocabulary and its suggestions offered limited support here because of the different inflected

[7]The language tagging on the SMS level was made at an earlier date.

[8]The average SMS counts 115 characters or 20 tokens, the longest one 2,374 characters or 425 tokens

forms. Looking back, the vocabulary lookup should probably have been implemented as a fuzzy one, so as to also find similar forms.

6 Conclusions

Manual glossing of a non-standard corpus is a labor-intensive job. Providing a tool adjusted to the specific needs, thus allowing for cooperation and an optimized workflow, is imperative, not only because it speeds up the process, but also because it supports consistency. The technology used in our project fulfilled its task, but results could have been improved with a fuzzy vocabulary search and a yet more intensive monitoring of the glossing process.

References

[1] Alistair Baron and Paul Rayson. VARD 2: A tool for dealing with spelling variation in historical corpora. *Proceedings of the Postgraduate Conference in Corpus Linguistics, Aston University, 2008*. eprints.lancs.ac.uk/41666/1/BaronRaysonAston2008.pdf

[2] Douglas Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. Internet Engineering Task Force, RFC 4627, 2006. www.ietf.org/rfc/rfc4627.txt

[3] Christa Dürscheid and Elisabeth Stark. sms4science: An International Corpus-Based Texting Project and the Specific Challenges for Multilingual Switzerland. In Crispin Thurlow and Kristine Mroczek, editors, *Digital Discourse: Language in the New Media.*, pages 299–320. Oxford University Press, New York, London, 2011.

[4] XML Path Language (XPath) 2.0 (Second Edition). *W3C Recommendation 14 December 2010*. www.w3.org/TR/xpath20/

[5] XQuery Update Facility 1.0. *W3C Recommendation 17 March 2011*. www.w3.org/TR/xquery-update-10/