Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022)
Distributed under Creative Commons License Attribution 4.0 International (CC BY 4.0)

Lemma Hunting: Automatic Spelling Normalization for CMC Corpora

Eckhard Bick

University of Southern Denmark

eckhard.bick@mail.dk

Abstract

This paper presents and evaluates a method for automatic orthographic normalization and the treatment of out-of-vocabulary words (OOV) in German social media data. The system uses a cascade of spellchecking operations including casing-, sound- and keyboard-based letter permutations, as well as letter context likelihoods, and combines partial and root spellchecking with compound analysis and heuristic inflection analysis in novel ways. The system also handles contractions, elisions and some tokenization errors. In addition, pattern-based recognition of foreign words and abbreviations is attempted, supported by jargon-informed lexicon expansion. Contextual Constraint Grammar (CG) disambiguation is used to resolve possible ambiguity. For Twitter data, F-scores of 87.3 and 77.1 were achieved for the identification and correct lemmatization, respectively, of German spelling errors and non-standard abbreviations. 77.6% of foreign words were recognized with 86.5% precision and 1/3 POS errors.

1 Introduction

Computer-mediated communication (CMC) is a notoriously difficult genre to annotate, an important issue being non-standard orthography and unusual word formation. For Social Media, in particular, Proisl (2018) and Beißwenger (2016) mention a host of problems such as out-of-vocabulary words (OOV), emoticons/emojis, interaction words (*lach* [laugh], *heul* [cry]), URL's and discourse links (hashtags and user id's), onomatopoeia, spelling variation and contractions, emphasis by upper-casing or letter repetition, as well as syntactic idiosyncrasies. In a corpus annotation scenario, all of these may lead to reduced lexicon coverage, affecting tagging performance. Thus, Neunerdt (2013) reports a drop in accuracy from 95.8% to 68% for OOV words, a problem he successfully tackled by adding a specialized web lexicon. But even with word additions and a correct (heuristic) POS assignment, a failure to group spelling variations, abbreviations and spelling errors under the same lemma negatively affects the possibility of corpus searches and statistics. In this paper, following Sidarenka et al. (2013), we suggest an automatic, spellchecking-like normalization process to address the problem, providing a common lemma for spelling variants and outright errors at the same time. For a language like German, compound analysis may also increase the search-accessibility of a corpus, and prevent false positive spelling corrections. The work presented here was performed on a large German Twitter and Facebook corpus compiled for the XPEROHS hate speech project (Baumgarten et al. 2019) and annotated with a multi-level Constraint Grammar (CG) parser (GerGram[1]). All examples in the paper are authentic exerpts from this corpus.

2 Systematic normalization

A relatively straightforward first step of

[1] https://visl.sdu.dk/de/parsing/automatic/

normalization concerns systematic variation, especially re-casing of lower-cased German nouns and of words written in all-uppercase for emphasis, both very common in our corpus. However, ignoring upper/lower case may lead to ambiguity between two German words or a foreign and a German word. This needs to be resolved contextually and is a possible source of errors.

Another case of systematic variation is gendering, which in German writing manifests as a female suffix, *-In* (sg.) or *-Innen* (pl.), attached to the (male/neutral) root with a variety of separators ('*', '_', '/' or '#') or with only the upper-case 'I' as a separator. For word classification and corpus search purposes all should be grouped under one lemma. Sometimes, this task borders on spellchecking or lexicography. Thus, our corpus contained examples of plural or adjective roots (*Freund<u>e</u>Innen* 'friends', *GrünInnen* 'Green Party-ists') and phonetic e-ellision (*RabaukInnen* - *Rabauke* 'brawler').

3 Spell-checking techniques

The second, and more challenging, step in normalization consists of spell-checking proper. In a text processor environment, a spell-checker offers a prioritized list of suggestions to be interactively processed by a human user. For automatic spell-checking, this is obviously not possible, so we only allowed suggestions with a Levenshtein distance of 1, meaning that the correction can be achieved by substituting, inserting or deleting a single letter. Again, contextual disambiguation may be necessary, because even at the Levenshtein-1 level, more than one correction may be possible. In our setup, disambiguation is an automatic side affect of ordinary CG disambiguation, triggered by differences in POS or inflection between the possible corrections.

To validate letter changes as legitimate corrections, we use a fullform dictionary with 1.23 million correct entries, consisting in part of a proof-read token list from non-CMC corpora, in part of fullform expansions arrived at by using German inflectional paradigms. The dictionary also contains 68907 error forms with their correction(s), also these consisting of manually sanctioned corpus examples and some paradigmatic expansion. The lexicalized error forms complement free spell-checking in two ways: First, in obvious cases, they can pre-empt the need for contextual disambiguation. Second, they represent a safe option for covering cases with higher Levenshtein-distance above 1.

Our spell-checking pipeline consists of a cascade of steps progressing from safe to unsafe. The first round mostly contains letter changes sanctioned by phonetic similarity, QWERTY keyboard layout or surrounding letters[2]. This module is run after ordinary lookup, inflectional analysis and prefix-/suffix-stripping, but before compound analysis. It performs the following checks:

- keyboard adjacency (e.g. *v/b, b/n*) or left-right confusion (e.g. *s/l*)
- phonetics, e.g. vowel lengthening markers (*i/ie/ih, versö(h)nlich*) and other grapheme ambiguity (*äu/eu*) or silent consonants (*ck/k, tz/z, ch/sch*)
- s-errors and pre-reform spelling (*ss/ß*)
- umlaut / diacritics (e.g. *u/ue/ü, a/ae/ä, o/oe/ö, e/é*)
- gemination errors and letter repetition (*Pa<u>p</u>kasse, Ta<u>nnn</u>te, g<u>aaa</u>nz lang)*
- weak letter omission: *g(e)kauft, bedeuten(d)ste, pakistan(i)schen*
- extra letter: *Bein(e)ame, Freundin(g)*
- letter pair repetition: *Ahnen(en)reihe, digit(it)ale*
- letter swap: *turg->trug, gignen->gingen*

It is a specific trait of German that a large proportion of OOV words are ordinary

[2] In this module, change patterns may involve 2 changed letters, or unchanged letters, and in that sense, while safer, are not ordinary Levenshtein-1 spellchecks.

compounds[3]. Further spellchecking is therefore blocked if morphological analysis can identify a high-confidence compound split, based on lexicon support for both parts, as well as their length, POS and semantics.

When spell-checking is activated, it is carried out by a letter-permutation subroutine. The task of trying out all possible letter changes and comparing them to the lexicon is surprisingly complex: For the average 6-letter word there are 5 swaps, 6 deletions, 5 splits, 25 * 6 substitutions and 26 * 6 insertions, resulting in 322 look-ups. Many of these may match a real word and need to be prioritized. We use a letter-context frequency strategy[4] to address both the complexity and the prioritization issue. For this, we extracted letter quintuples from corpus data, counting space as a letter, too, and computed the letter likelihood for the three middle positions given their left and right letter neighbors in the quintuple. These data can be used to suggest the most likely substitution or insertion, rather than trying them all with no prioritization. The overall worth of a possible correction word is then computed as the product of its normalized corpus frequency and either a fixed "method prioritization constant" (for swaps and deletions) or the frequency of a given substitution or insertion relative to the embedding quintuple. Finally, the subroutine will return the correction operation with the highest value, considering only corrections that can be verified in the fullform lexicon.

In order to minimize false hits, the letter-permutation subroutine is first fed unknown word parts of partially recognized words, reserving full-word spellchecking as a last step. For this purpose, the system remembers "almost"-hits in the compound analysis of longer words, where a first or second part could be matched in the lexicon, but the remainder of the word (i.e. the potential other compound part) could not. In these cases, if both parts have a minimum length, the unknown part is spellchecked on its own:

pædophlie|verdächtig > pädophilieverdächtig
Voraussage|mögichkeit >Voraussagemöglichkeit

Failing this, the system looks up the last 5 letters in an endings/affix database, and spellchecks the remainder as a kind of artificial root. Only after this, as a last resort, fullform spellchecking is carried out. To avoid over-generation in the face of short word parts, letter deletions are not allowed for compound parts, and splitting is only allowed for full words.

4 Word splitting and fusion

A certain amount of spelling variation can not be addressed with the above techniques, because they concern tokenization. The most common problems were English-style splitting of noun compounds (e.g. *Terroristen Pack, Kanaken Gang*) and colloquial contractions of pronouns and short verbs (e.g. *machen wirs [=wir es] doch* ['let's do it'], *kannste [= kannst du]* ['can you']). We use lexical rules to split the contractions, maintaining the fullform on the first part and marking the split on both parts. For identifying split compounds (in particular, OOV compounds), contextual CG rules are necessary, implying a certain risk of error. Rather than creating a new, fused token, we mark the split on the first part, but maintain both as tokens in order to preserve the individual lemmas, as well as semantic and other tagging, for corpus searching purposes.

5 Abbreviations and foreign words

Abbreviations are at the same time a very frequent and a very variable feature of CMC data. Thus, neither casing nor the presence and

[3] Our corpus contained 10% compounds, of which 1/6 were OOV, i.e. found through live analysis. 2/3 of the OOV compounds were flagged as hight confidence. 17% of low-confidence compounds were really names or spelling errors.

[4] The size of the context window has to be balanced to avoid sparse-data problems, but in prinicple, a similar strategy could be used for entire words and word contexts of sufficient frequency (future work). Also, the list of correction possibilities could be passed on to CG disambiguation, exploiting the wider context of the sentence/utterance. However, while the latter technique worked well for ordinary, interactive spell-checking, it proved to be much less safe for cases where the context itself is also full of errors, orthographical creativity and OOV tokens, as is often the case in CMC data.

placement of dots can be trusted. For instance, *zB, zB., z.B., z.b.* all mean *zum Beispiel* ('for instance'). There is also great variation as to which letters (other than the first) are used to abbreviate single words (*vll, vllt, vlt = vielleicht* ('maybe'). Very typical are multi word expressions (MWEs) representing small utterances, e.g. *ka = keine Ahnung* ('no idea') or *kb = kein Bock* ('no desire to'), including many English ones, e.g. *WTF (what the fuck)* or *omg (oh my God).* Arguably, recognizing abbreviations is not a classical spellchecking, but either a lemmatization/normalization task (for *z.B.* and *vlt*) or a lexicalization task (*WTF, omg*) necessary for assigning a "syntactically harmless" word class such as adverb or interjection, but also to *prevent* spellchecking an abbreviation into a regular word (e.g. *omg* as *mg* or *Oma*). Foreign words need to be recognized for the same reason, also if they are not abbreviated, because a small change may make them look like a German word. The problem was addressed by pre-filtering input lines that looked English in their entirety, by matching certain letter patterns typical of English but not of German, and by adding some genre-typical words may to the lexicon.

6 Evaluation

We evaluated the performance of the normalizer tool on two chunks of tweets from a random day. The sample consisted of 5764 tokens containing 4761 words when excluding punctuation, web links and @-names. Of these, 6.5% were words in need of spelling correction and/or other lexical normalization[5] to support a correct reading[6]. Another 2.1% were non-name foreign[7] words also representing a recognition challenge. The system identified 82.5% of the spelling errors and non-standard abbreviations, and 77.6% of the foreign words as such. 66.8% of the former (79.9% of the recognized ones) were assigned the correct normalization/lemma. Of the unrecognized spelling errors, half were OOV, half were real word errors, e.g. *frage* not recognized as the noun *Frage,* but rather accepted as a possible (but wrong in-context) inflection form of the verb *fragen. 7.2*% of all words marked as spelling errors were false positives, mostly foreign words misread or, sometimes, miscorrected as German, e.g. *locker* (a German adjective, but in-context an English noun) or *freefall* (read as *Freifall*). These numbers translate into F-scores of 87.3 and 77.1 for the identification and correction of spelling errors, respectively (see Table 1).

	R	**P**	**F[8]**	**ERR[9]**
identification task	82.5	92.8	87.3	77.1
correction task	66.8	91.2	77.1	60.3
foreign word recog.	77.6	85.4	81.3	64.3

Table 1: Recall, precision, F-score (%), ERR

The ERR score for the correction task can in principle be compared to results obtained in the shared task for multilingual lexical normalization (MultiLexNorm) in the W-NUT workshop 2021 (van der Goot et al., 2021), where only the best system, ÚFAL (Samuel and Stracka, 2021), achieved a higher score (ERR=66.2) in the intrinsic evaluation. However, the data sets are not directly comparable, and differences in normalization principles and tokenization made it impossible to perform a true cross-evaluation within the scope of this paper[10].

Recognition of foreign words worked reasonably, but not as well as German normalization, considering that 1/3 of the recognized foreign words received a wrong POS. 7% of the non-name foreign words were tagged as proper nouns because they were in upper case. For foreign words, false positives were triggered by lower-case names or by some OOV

[5] The latter includes e.g. clitic-splitting and recognition of chat-style abbreviations and interjections, that would otherwise be OOV and/or get a wrong lemma or word class.

[6] A further 0.5% of minor errors were ignored, These were errors concerning hyphenation and inflection not causing POS changes or lemmatization errors.

[7] Counting foreign words occurring in German sentences. Six separate short sentences (4 English, 2 Spanish) with 5-6 words each, were not included here.

[8] F1-score, defined as 2*recall*precision/(recall+precision)

[9] Defined as ERR=(CF-FP)/(CF+FN), with CF=correctly found, FN=false negatives, FP=false positives

[10] Still, as a first step, a filter program was written to convert system output into the MultiLexNorm two-column format.

abbreviations without dot, e.g. *guna* (= *Gute Nacht* 'good night').

7 Conclusions and outlook

We have discussed a method for ameliorating the high OOV rate in German CMC data using automatic spellchecking, morphological analysis and letter pattern recognition. The system has been integrated with a CG disambiguator and parser, and used in the annotation of a 3-billion-word Twitter corpus with satisfactory results. Based on qualitative error analysis from the test run, real-word errors should also be addressed, in particular where lower-casing errors of real German words can be confused with other German words, foreign words or abbreviations. For this task, wider word context should be exploited, either statistically and/or through CG disambiguation of the most likely replacements.

References

Baumgarten, N.; Bick, E.; Geyer, K.; Iversen D. A.; Kleene, A.; Lindø, A. V.; Neitsch, J.; Niebuhr, O.; Nielsen, R.; Petersen, E. N. 2019. Towards Balance and Boundaries in Public Discourse: Expressing and Perceiving Online Hate Speech (XPEROHS). In: Mey, J., Holsting, A., Johannessen, C. (ed.): RASK - International Journal of Language and Communication. Vol. 50., pp. 87-108. University of Southern Denmark.

Beißwenger, M.; Bartsch, S.; Evert, S.;Würzner, K.-M. 2016. EmpiriST 2015: A shared task on the automatic linguistic annotation of computer-mediated communication and web corpora. In: Paul Cook et al. (ed.): Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task. pp. 44-56. Berlin: Association for Computational Linguistics.

Bick, E.; Didriksen, T. 2015. CG-3 – Beyond Classical Constraint Grammar. In: Beáta Megyesi: *Proceedings of NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania*. pp. 31-39. Linköping: LiU Electronic Press. ISBN 978-91-7519-098-3

Neunerdt, M., Trevisan, B.,Reyer, M., Mathar, R. Part-of-Speech Tagging for Social Media Texts. (2013). Language Processing and Knowledge in the Web. pp. 139-150. Springer

Proisl, T. SoMeWeTa: A Part-of-Speech Tagger for German Social Media and Web Texts. In: Proceedings of ELREC 2018. pp. 665-670

Samuel, D. and Straka, M. 2021. ÚFAL at Multi-LexNorm 2021: Improving multilingual lexical normalization by fine-tuning ByT5. In Proceedings of the 7th Workshop on Noisy User-generated Text (W-NUT 2021), Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sidarenka, U., Scheffler, T., Stede, M. Rule-Based Normalization of German Twitter Messages. (2013). In: Proceedings of the GSCL Workshop: Verarbeitung und Annotation von Sprachdaten aus Genres internetbasierter Kommunikation.

van der Goot, R. et al. 2021. MultiLexNorm: A Shared Task on Multilingual Lexical Normalization. In: Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021). pp. 493–509, Online. Association for Computational Linguistics.