Truecasing German user-generated conversational text

Yulia Grishina **Thomas Gueudré** **Ralf Winkler**
Amazon Alexa
{yuliag,tgueudre,rwinkle}@amazon.com

Abstract

True-casing, the task of restoring proper case to (generally) lower case input, is important in downstream tasks and for screen display. In this paper, we investigate truecasing as an intrinsic task and present several experiments on noisy user queries to a voice-controlled dialog system. In particular, we compare a rule-based, an n-gram language model (LM) and a recurrent neural network (RNN) approaches, evaluating the results on a German Q&A corpus and reporting accuracy for different case categories. We show that while RNNs reach higher accuracy especially on large datasets, character n-gram models with interpolation are still competitive, in particular on mixed-case words where their fall-back mechanisms come into play.

1 Introduction

Truecasing is a natural language processing task that consists of assigning proper case to all words within a text where such information is not available. For many natural language applications, it is an important pre-processing step, shown to be useful in downstream tasks such as named entity recognition (Bodapati et al., 2019), automatic content extraction (Cucerzan, 2010) and machine translation (Etchegoyhen and Gete, 2020). For languages with complex casing rules (e.g., German), determining part of speech (POS) or named entity becomes almost impossible without casing information:

(1) Was fressen **Fliegen**?
what eat flies

(2) Wie hoch **fliegen** Vögel?
how high fly birds

In this example, the token 'fliegen' in (1) is a noun and therefore should begin with a capital letter, while in (2) it is a verb and is hence lowercased.

Truecasing is particularly hard in the German language, as it follows multiple capitalization rules, some of which we describe below. For instance, all nouns (including both common nouns and proper nouns) and nominalized verbs must be capitalized. Furthermore, German adjectives should be capitalized when used as substantive (e.g., *bei Grün*), while not being confused with their true adjective form (which is not capitalized). Another common ambiguity is the formal pronoun *Sie*, capitalized in all of its form (as opposed to informal *sie* which must be lowercased).

Truecasing becomes particularly important in dialog systems that aim to correctly interpret the natural language of user queries. Such systems often rely on automatic speech recognition (ASR) output, often quite noisy and non-grammatical as users frequently engage with their devices using highly colloquial speech. This output is used by downstream modules such as spoken language understanding (SLU) or entity resolution (ER). Another use case is displaying such output, e.g., on a device with screen, where casing information is necessary for readability and customer experience.

In this paper, we investigate different approaches to truecasing German user queries in the context of a voice-controlled dialog agent. We focus on informational and world knowledge queries, in which the users ask the device questions about, e.g., recent events, famous people, general world knowledge etc. In particular, we pit a rule-based approach against two learnt models, a character n-gram one and a neural one. We report the results across different case categories and discuss the benefits and the drawbacks for each of the approaches.

This paper is organized as follows: In Section 2, we provide an overview of related work on text truecasing. Next, we describe our experiment (Section 3), giving the overview of the truecasing approaches and datasets used. In Section 4, we

Proceedings of the 2020 EMNLP Workshop W-NUT: The Sixth Workshop on Noisy User-generated Text, pages 143–148

present the results of the experiment and provide a detailed error analysis. Finally, we discuss the results and outline future directions of research (Section 5).

2 Related work

The simplest approach to truecasing is to use unigram tagging, converting all tokens to their most frequent form in the training data. It is typically used as a baseline in capitalisation experiments which improve over this straightforward approach (e.g. (Lita et al., 2003; Chelba and Acero, 2006)).

Most of the earlier approaches to truecasing rely on statistical modeling and work mostly at word level. For instance, one of such approaches was proposed by Lita et al. (2003) who used a trigram language model and sentence level context to predict the most probable case sequence in a sentence. Next, Wang et al. (2006) proposed a CRF-based probabilistic model, which exploited bilingual information and was able to improve over monolingual methods on machine translation output, while Chelba and Acero (2006), Batista et al. (2008) used an approach based on maximum entropy models. Benefiting from larger computational power, the more recent approaches advocate modelling at character level. Susanto et al. (2016) first proposed to use a character-level RNN for truecasing, demonstrating the superiority of a character-level approach to word-based approaches. They show that the benefits of such approaches over word-based are that they are able to better generalise on "unseen" words and they also perform better on mixed-case[1]. Following this argument, we compare their architecture with based large character n-gram models with various smoothing.

3 Experiments

3.1 Dataset

For our experiments, we rely on two different datasets: (a) an internal German Q&A dataset, and (b) the German part of the Leipzig corpus collection[2] (Goldhahn et al., 2012). The Q&A dataset is a corpus that consists of frequent questions coming from a spoken dialog system users and answers provided by the same system[3]. It contains 30K sentences, or 410K tokens, covering mostly the Information domain. All of the data used in this paper was anonymized prior to the experiment, to make sure that it does not include any user identifying information. A typical example from the corpus would be the following query:

(3) [User:] Welche Farben hat der Regenbogen? ('What is the color of the rainbow?')

[Device]: Die Regenbogenfarben sind Rot, Orange, Gelb, Grün, Hellblau, Dunkelblau und Violett. ('The rainbow colors are Red, Orange, Yellow, Green, Blue, Indigo, and Violet.')

Leipzig Corpus collection is a freely available corpus of newspaper articles. For our experiment, we have taken a cleaned dataset consisting of newspaper texts of 2015[4], from which we have randomly selected 100K sentences (around 2M tokens) for the experiment.

For both datasets, we used a 80/20 random split for training and test purposes, and use the same split for both the n-gram and neural language modelling approaches. A small portion (5%) of the train set was kept as development set, to tune the hyper-parameters of each model.

3.2 Truecasing approaches

3.2.1 Majority rule with tagged LM as baseline

We provide a simple baseline, using the majority rule and a tag-based model. The majority rule consists in turning each token of the test-set into the most common capitalized form in the training set. Words that have not been seen in the training corpora are left untouched and tallies are broken by lexicographic order (therefore picking the lowercased form first).

As a standard way of taking into account sequence correlations, we tag each training example with the case category (upper, lower, mixed and punct)[5] and train a 10-gram LM over the tags sequences, built with Kneser-Ney interpolation method. Finally, we compose this LM with the above majority baseline.

[1] Mixed-case words are those that contain upper-cased characters in the middle of a word, such as 'Rheinland-Pfalz'.

[2] https://corpora.uni-leipzig.de?corpusId=deu_newscrawl-public_2018

[3] The Q&A data is a subset of the questions and answers that can be accessed via https://alexaanswers.amazon.de.

[4] https://www.kaggle.com/rtatman/3-million-german-sentences

[5] "Upper" contains tokens that have a capital letter only at the beginning, while "lower" means that all letters in a token are lowercased. "Mixed" category includes tokens that have capital letters in the middle of a token as well as acronyms (e.g., *NASA*).

	Q&A				Leipzig			
	ΔAcc	ΔP	ΔR	$\Delta F1$	ΔAcc	ΔP	ΔR	$\Delta F1$
Baseline Majority rule	-8,98	-0,03	-26,47	-17,08	-3,50	+0,28	-11,13	-6,78
Rule-based	-2.43	-8.74	+0.62	-3.82	-4.31	-18.07	+3.88	-6.76
Kneser-Ney 5-gram leipz	-1.24	-7.74	+2.37	-2.45	+0.07	-7.31	+7.80	+0.78
Kneser-Ney 10-gram leipz	+0.11	-5.90	+4.30	-0.56	+3.00	-2.14	+11.93	+5.44
Kneser-Ney 15-gram leipz	+0.19	-5.81	+4.35	-0.49	+3.15	-2.01	+12.29	+5.68
Kneser-Ney 5-gram q&a	+2.08	-2.37	+8.08	+3.09				
Kneser-Ney 10-gram q&a	+4.09	+1.15	+10.4	+6.01				
Kneser-Ney 15-gram q&a	+4.20	+1.25	+10.64	+6.19				
char-rnn LSTM leipz	+1.43	-3.15	+5.21	+1.27	+4.29	+0.18	+13.52	**+7.39**
char-rnn GRU leipz	+1.26	-3.76	+5.13	+0.92	+4.13	-0.33	+13.27	+7.01
char-rnn LSTM q&a	+4.28	+1.22	+10.58	+6.14				
char-rnn GRU q&a	+3.79	+0.56	+9.39	+5.21				
Kneser-Ney 15-gram q&a + leipz	+4.58	+1.85	+11.04	+6.68	+3.14	-1.99	+12.26	+5.67
char-rnn LSTM q&a + leipz	+4.81	+2.17	+11.28	**+6.96**	+3.55	-1.21	+12.21	+6.04

Table 1: Truecasing results (%) relative to the tag-based LM baseline: Accuracy, Precision, Recall and F1 across all approaches evaluated on two datasets - the Q&A corpus (column 2) and the Leipzig corpus (column 3). For the n-gram LM and the RNN approaches, we also specify the corpus used for training. Majority rule baseline was trained and evaluated on the same dataset (no cross-domain evaluation). Models trained on the Q&A corpus were not evaluated on the Leipzig corpus.

3.2.2 Rule-based approach

For this approach we compiled grammars based on regular expressions into weighted finite-state transducers (FSTs) using OpenGrm and Thrax libraries (Roark et al., 2012). The system consists of token lists, each associated with a POS (e.g., pronouns, adjectives, verbs, etc.) and several rules containing POS patterns frequently preceding tokens in uppercase. If an input token matches one of the lists, a path with decreased local weight is generated and the following token, if not in one of the lists, is uppercased. Each additional match within a rule adds a further decreased local weight, reflecting the assumption, that the probability for an token in uppercase increases with the number of known predecessors. By applying these rules on input text, we end up with several possible paths for each sentence, and select the path with lowest global weight as best candidate. Rules were hand-tuned on a development subset of the Q&A dataset.

3.2.3 N-gram language modelling with FSTs

This approach also relies on the FST machinery, but makes use of a trained language model (LM) to re-rank the possible hypothesis. The approach is standard (Manning et al. (2008)), except from the fact we employ it with characters rather than words.

We first build an FST that, when composed with a lower-cased input, returns a lattice of all possible capitalized variants (for example, “hi” being transformed to “Hi”, “hI” and “HI”).

To estimate the probability of each hypothesis produced by the above lattice, we collect N-gram counts from the capitalized training corpus and re-normalize them into probabilities. Especially for high N values, N-gram models are sparse (as some sequences of characters are seen only a few times) and require some back-off or smoothing strategies. We pick the Kneser-Ney interpolation scheme (using the OpenGrm library), a widely used scheme which estimates the probability of a given N-gram based on lower order statistics (Ney et al. (1994)).

Finally, model prediction is obtained by composing the lattice of capitalized variant with the LM, and extracting the path of lowest cost.

We tried models with various smoothing schemes (Katz, Witten-Bell and Kneser-Ney) on the dev set, and kept the interpolated Kneser-Ney for its superior accuracy. For completeness, we also report the results for different n-gram sizes (5, 10 and 15) and observe that larger values lead to higher accuracy, albeit greatly increasing the required memory (1Gb for the largest models).

	Rule	KN 15-gram q&a	KN 15-gram leipz	char-rnn q&a	char-rnn leipz
Upper	90.2	97.0	95.3	97.3	96.9
Lower	92.6	98.6	97.8	98.6	98.9
Mixed	68.1	94.0	84.9	89.6	75.6

Table 2: Token accuracy per case category (upper, lower, mixed case)

	q&a	leipz
Upper	21 691	101 177
Lower	46 171	245 225
Mixed	6947	23 957

Table 3: Number of tokens with upper, lower, mixed case per test set

3.2.4 Recurrent neural networks

We used the approach of Susanto et al. (2016) to train a character-level RNN language model. We use the provided implementation[6] to train a small RNN model with 2 layers and 300 hidden states, varying the type of the hidden unit (LSTM/GRU). It uses truncated backpropagation for 50 time steps. After training, the model with the smallest validation loss after 30 epochs is chosen.

4 Results and error analysis

In the following, we present a comparison of the approaches on the the Q&A corpus and the Leipzig corpus. Improvements for each of the approaches relative to the tag-based LM baseline are presented in Table 1. We are also reporting results on a mixed dataset, i.e., taking all available Q&A training data, and adding a similar amount of the Leipzig corpus training data. While we are reporting relative improvement results due to privacy concerns on the Q&A dataset, baselines on this task are known to be quite strong (over 90% accuracy, e.g. (Lita et al., 2003)) and state-of-the art approaches such as the neural networks used in this paper reach accuracy of the order of 96-98% (Susanto et al., 2016).

4.1 Case categories and mixed-case words

First, we look at accuracy per case category (upper, lower, mixed), which can be seen in Table 2. The overall number of tokens belonging to each of the categories are presented in Table 3.

As one can see from Table 2, mixed case words are best handled by the n-gram LMs, while the rule-based approach shows the lowest accuracy on those. Low scores on mixed case words for the Leipzig corpus are due to a large portion of proper names coming from newspaper articles (e.g., "NewVoiceMedia" or "TecDAX") that were not capitalized properly. The success of n-gram LMs for mixed cases is naturally explained by the back-off smoothing mechanism in interpolated n-gram models, where subwords found capitalized in the corpora largely contribute to the final cost.

4.2 Unseen words

From the test/train splits, there are 16048 unseen words in the Leipzig corpora, and 3494 in the Q&A corpora. Accuracy percentages for those unseen words are presented in Table 4 (we do not evaluate unseen words on the rule-based approach, as we used automatically created POS lists which therefore may have contained unseen words). The RNN approach in this case is the most accurate, following the conclusion of Susanto et al. (2016).

We attribute the bulk of the mistakes of the n-gram model to its inability to capture longer dependencies. Despite a large n-gram value (15), the model is often unable to implicitly obtain the POS of the token when the information is contained in the suffix. The model then defaults to the most common occurrence. For example, the noun "Gleichstrom" ("direct current") is normalized as "gleichstrom" because of the commonly seen adjective "gleich" ("equal"), without considering the suffix "strom".

4.3 Accuracy by Part of Speech

We used spacy[7] German model to PoS tag our test corpus, and subsequently extracted most common categories of errors. Unsurprisingly, common nouns, adjectives and proper nouns constituted the highest proportion of errors in all approaches. We inspected the most frequent failing tokens for each

[6] https://gitlab.com/raymondhs/char-rnn-truecase

[7] https://spacy.io/

	Unseen words
KN 15-gram q&a	76.00
KN 15-gram leipz	78.06
char-rnn LSTM q&a	86.10
char-rnn LSTM leipz	83.20

Table 4: Token accuracy for unseen words on the Q&A corpus

	Sentence-based accuracy
KN 15-gram leipz	65.45
char-rnn LSTM leipz	72.60
char-rnn GRU leipz	70.00

Table 5: Sentence-based accuracy on Leipzig corpus

of the approaches on the Q&A dataset, but the models show no major difference there. For all of them, the failing tokens belong to the class of frequent pronouns or prepositions, whose capitalization is particularly ambiguous and context dependent, such as "Sie" or "die".

4.4 Sentence-based accuracy

We report the sentence accuracy for the LM and RNN approaches as well, in Table 5. Sentence-based accuracy is computed as the ratio of the number of sentences where all tokens were predicted correctly to the overall number of sentences in the test set. Sentence-based accuracy for the majority rule baseline is extremely low (<15.0) which is mostly due to the fact that the initial letter in a sentence is frequently left lowercased.

We distinctly observe that RNNs outperform simple n-gram LMs on this conservative metric. This implies that RNNs often perform a flawless normalization of the whole sentence, while n-gram approaches scatter mistakes more uniformly across sentences. However, a more detailed analysis shows that mistakes done by RNN are more critical. For instance, it sometimes tends to overgenerate, which results in hardly readable mixed-cased (e.g., "KAffeMaschine") or conversational words (e.g., capitalizing the colloquial form 'ne' of the indefinite article 'eine').

5 Conclusion and future work

In this paper, we presented a study on truecasing, a common task in natural language processing, either as a pre-processing step in a larger pipeline (f.e. in MT), or as a post-processing one (f.e. when displaying the output of an spoken language system on screen). Comparing rule-based, n-gram and neural models, we showed that, while state of the art methods such as DNNs unsurprisingly reach the highest accuracy, standard n-gram models are still competitive, in particular on mixed-case words, where their fall-back mechanisms come into play. They also fair well in noisier corpora (such as the Q&A corpora).

Truecasing, as a part of text normalization, is peculiar in that its bulk can be solved simply by a few hand-written rules, with however a long tail of very difficult cases such as acronyms, unseen words. Finding a proper balance between the flexibility of neural approaches, and the controlled, more interpretable behaviour of FST-based systems, remains an open and challenging problem (Mansfield et al. (2019), Sproat and Jaitly (2016), Zhang et al. (2019)).

Acknowledgements

We thank Fernando Koch and Saskia Schuster from Amazon Alexa for providing the Q&A data, and Spyros Matsoukas for his valuable feedback on an earlier draft of this paper. We also thank anonymous reviewers for their helpful comments.

References

Fernando Batista, Nuno Mamede, and Isabel Trancoso. 2008. Language dynamics and capitalization using maximum entropy. In *Proceedings of ACL-08: HLT, Short Papers*, pages 1–4.

Sravan Bodapati, Hyokun Yun, and Yaser Al-Onaizan. 2019. Robustness to capitalization errors in named entity recognition. *arXiv preprint arXiv:1911.05241*.

Ciprian Chelba and Alex Acero. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Silviu Cucerzan. 2010. Does capitalization matter in web search? In *KDIR*, pages 302–306.

Thierry Etchegoyhen and Harritxu Gete. 2020. To case or not to case: Evaluating casing methods for neural machine translation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3752–3760.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *LREC*, volume 29, pages 31–43.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. Truecasing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 152–159.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

Courtney Mansfield, Ming Sun, Yuzong Liu, Ankur Gandhe, and Björn Hoffmeister. 2019. Neural text normalization with subword units. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 190–196, Minneapolis, Minnesota. Association for Computational Linguistics.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8(1):1–38.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66, Jeju Island, Korea. Association for Computational Linguistics.

Richard Sproat and Navdeep Jaitly. 2016. RNN approaches to text normalization: A challenge. *CoRR*, abs/1611.00068.

Raymond Hendy Susanto, Hai Leong Chieu, and Wei Lu. 2016. Learning to capitalize with character-level recurrent neural networks: an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2090–2095.

Wei Wang, Kevin Knight, and Daniel Marcu. 2006. Capitalizing machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 1–8.

Hao Zhang, Richard Sproat, Axel H Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark. 2019. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337.