INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

A Mostly Data-driven Approach to Inverse Text Normalization

Ernest Pusateri[1], *Bharat Ram Ambati*[1], *Elizabeth Brooks*[1], *Ondrej Platek*[2], *Donald McAllaster*[1], *Venki Nagesha*[1]

[1]Apple Inc., USA
[2]Charles University, Czech Republic
{epusateri,bambati}@apple.com

Abstract

For an automatic speech recognition system to produce sensibly formatted, readable output, the spoken-form token sequence produced by the core speech recognizer must be converted to a written-form string. This process is known as inverse text normalization (ITN). Here we present a mostly data-driven ITN system that leverages a set of simple rules and a few hand-crafted grammars to cast ITN as a labeling problem. To this labeling problem, we apply a compact bi-directional LSTM. We show that the approach performs well using practical amounts of training data.

Index Terms: speech recognition, inverse text normalization, LSTM

1. Introduction

Inverse text normalization (ITN) is the process of taking a token sequence in spoken form produced by an automatic speech recognizer and converting it to a written form suitable for presentation to users and processing by downstream components. The entities requiring significant transformation to go from spoken form to written form include cardinals and ordinals as well as more complex items like dates, times and addresses. In [1] the written forms of such entities are called non-standard words (NSWs) and an NSW taxonomy is provided. Table 1 presents some examples of spoken-form input and written-form output typical of what we see in our system.

There has been little work published that directly addresses the ITN problem. Exceptions include [2, 3, 4]. In [3], hand-crafted grammars are used to over-generate written-form hypotheses based on the spoken-form token sequence. A class language model (LM) is applied to choose the most likely written-form hypothesis. In [4], the automatic speech recognition (ASR) LM is built in the written domain. A finite-state verbalization model, built from hand-crafted grammars, is used to transform the written form into spoken-form token sequences. The written domain LM and verbalization model are combined to create the ASR decoding graph. ITN is built into the decoding graph in [2] as well.

Table 1: *Examples of spoken-form input and desired written-form output.*

Spoken Form	Written Form
one forty one Dorchester Avenue Salem Massachusetts	141 Dorchester Ave., Salem, MA
set an alarm for five thirty p.m.	Set an alarm for 5:30 PM
add an appointment on September sixteenth twenty seventeen	Add an appointment on September 16, 2017
twenty percent of fifteen dollars seventy three	20% of $15.73
what is two hundred seven point three plus six	What is 207.3+6

There has also been relevant work on related tasks, including the inverse of the problem addressed here, text normalization (TN). In [1], TN is formulated as a multistage process including stages to segment written-form input, tag the segments and transform the tagged segments into spoken-from output. Various techniques are explored for each stage. In [5], statistical machine translation (SMT) techniques are applied to TN. In [6], various RNN architectures including an attention-based sequence-to-sequence model, are applied. In [7] SMT techniques are applied to the task of speaking-style transformation, and in [8], a phrase-based SMT model is used to transform SMS messages into more conventional written-form output. In [9], a method is presented to format and error-correct ASR transcriptions using a probabilistic model to determine when automatically learned 1-to-n replacements should be applied.

In addition to at least partly data-driven approaches, it is possible to build an effective ITN system entirely from hand-crafted rules. The reference system in this work is one such system.

The approach we present here uses simple hand-crafted rule tables and a few hand-crafted grammars to cast ITN as a labeling problem. To the labeling problem, we apply a bi-directional long short-term memory (LSTM) neural network, motivated by the fact that LSTMs have proven highly effective in a variety of similar tasks [10, 11, 12, 13, 14].

Compared to entirely rule-based systems, our approach has a few advantages. First, system behavior can be changed through modification of the training data, which requires less specialized knowledge than writing rules. Second, our approach has the potential to learn transformations that occur in varied contexts, which may be difficult to express as rules. Finally, reasonably accurate rule-based systems require building and maintaining large, complex language-specific rule files. In our approach, the language-specific configuration files and grammars are relatively small and simple. In contrast to [4], where ITN functionality is built into the decoding graph, our approach has the practical benefit of maintaining the boundary between the language modeling and ITN components.

Our approach is perhaps most conceptually similar to [3]. That too is a hybrid approach using both hand-crafted grammars and a statistical model. There the form of the statistical component is a class N-gram. At a high level, the approach we present here uses simpler grammars and a more powerful and compact statistical model. We should also note that [3] assumed that spoken form/written form pairs were not available for training, and so they relied entirely on written-form data to train their system. In this work, we make the assumption that spoken form/written form training data is available.

We could have chosen to apply a more general attention-based sequence-to-sequence model, as [6] evaluates for TN.

http://dx.doi.org/10.21437/Interspeech.2017-1274

However, as pointed out in that work, the space of potential errors that such a model can make is quite large, and egregious errors can fundamentally change the meaning of the output. In addition, the much simpler model underlying our approach is likely to be less data hungry, faster to train and faster to run inference on.

The remainder of this paper is organized as follows: We describe the details of our approach and model in Sections 2 and 3. Section 4 presents experiments and results.

2. Approach

We first make the observations that (1) in nearly all of the transformations made by ITN, there is an obvious correspondence between spoken-form tokens and segments of the written-form output, and (2) the written-form segments will usually be in the same order as their corresponding spoken-form tokens.

Motivated by those observations, we formulate the transformation from spoken form to written form as the following three step process. In the first step, a label is assigned to each spoken-form token. The label specifies a series of edits to perform to the token string to get its corresponding segment in the written-form output string. In the second step, for each token, in order, the edits specified by the label are applied and the results concatenated. Finally, post-processing grammars are applied to regions tagged in the second step. Table 2 gives an example of the intermediate and final outputs of this process for two spoken-form token sequences.

2.1. Label Definition

Each label has six fields:

- **Rewrite**: Indicates the string, if any, that replaces the token string. The default value represents leaving the token string unmodified. Built-in options include *Capitalize* and *Downcase*.
- **Prepend**: Indicates the string, if any, that should be prepended to the token string. The default value represents prepending nothing.
- **Append**: Indicates the string, if any, that should be appended to the token string. The default value represents appending nothing.
- **Space**: Indicates whether or not a space should be placed before the token string. The default value represents putting a space before the string.
- **Post Start**: Indicates whether the token represents the start of a region where a post-processing grammar should be applied. If so, specifies the post-processing grammar to apply.
- **Post End**: Indicates whether the token represents the end of a region where a post-processing grammar should be applied. If so, specifies the post-processing grammar to apply.

We construct a finite-state transducer (FST) for each option for each field. To apply a label to the spoken-form token string, the corresponding FSTs are applied in sequence. The FSTs are constructed from Thrax [15] grammars. These include grammars for built-in options (e.g. the *Capitalize* rewrite option) as well as configurable grammars. Configurable Thrax grammars are automatically generated from tables. Table 3 shows excerpts from the rewrite table, and Table 4 is the append table.

2.2. Post Processing

For most transformations, label application is sufficient to produce the final written-form output. (See the first example in Table 2.) For the few phenomena that cannot be handled this way, we have a mechanism for applying post-processing grammars. When one of these grammars is to be used, label application will tag a region as requiring post processing, based on the *Post Start* and *Post End* fields. The appropriate grammar will then be applied to the region. Post-processing grammars are written in Thrax and compiled into FSTs.

In the systems we present here, we use post-processing grammars for currency expressions, relative time expressions and roman numerals. Currency grammars handle the reordering of the currency amount and the currency symbol, as shown in the second example in Table 2. The relative time expression grammar is used for spoken-form token sequences like "ten minutes to four", which must produce the written-form output "3:50". Although roman numeral formatting could theoretically be accomplished with the labeling mechanism, we use post-processing grammars, as it is straightforward to create a language-independent grammar converting cardinal numbers to roman numerals.

2.3. Label Inference

Our training data consists of spoken form/written form pairs. However, to train our model, we need spoken form/label sequence pairs. We use an FST-based approach to infer label sequences from spoken form/written form pairs. To support this process, we construct the following FSTs:

- E: An expander FST, which takes as its input a spoken-form token sequence and produces outputs where each token in the sequence has been prepended with every possible label.
- A_f: For each label field, f, an applier FST that takes a spoken-form token sequence with prepended labels and applies the action indicated by that field in each token's prepended label. Constructed from the FSTs built for the field options.
- R: A renderer FST, which takes as its input a sequence of processed tokens with labels prepended and strips the labels to produce written-form output, possibly with regions tagged for post-processing.
- P: A post-processor FST, which applies appropriate post-processing grammars to tagged regions and removes the tags. Constructed from the post-processing grammar FSTs.

To get the compatible set of spoken-form token sequences with prepended labels for a spoken form/written form pair, we perform the following sequence of operations, where S and W are FSTs constructed from the spoken-form token sequence and written-form output string. The $Proj_{in}$ operation copies each arc's input label to its output label, while $Proj_{out}$ does the opposite.

$$\begin{aligned} S_{exp} &= Proj_{out}(S \circ E) \\ W_{proc} &= S_{exp} \circ A_{Rewrite} \circ ... \circ A_{PostEnd} \\ S_{labeled} &= Proj_{in}(W_{proc} \circ R \circ P \circ W) \end{aligned} \quad (1)$$

From $S_{labeled}$, we can easily extract label sequences. For a vast majority of utterances (over 99% in our data), $S_{labeled}$ will consist of a single path. For those cases where it does not,

Table 2: *Two spoken-form token sequences with corresponding labels, output after label application and final written-form output after application of post-processing grammars.*

Spoken Form	Label						After Label Application	Written Form
	Rewrite	Prepend	Append	Space	Post Start	Post End		
February	-	-	-	-	-	-	February	February
twentieth	OrdinalAsCardinalDecade	-	Comma	-	-	-	20,	20,
twenty	CardinalDecade	-	-	-	-	-	20	20
seventeen	CardinalTeen	-	-	No	-	-	17	17
twenty	CardinalDecade	-	-	-	-	-	20	20
percent	PercentSign	-	-	No	-	-	%	%
of	-	-	-	-	-	-	of	of
two	Cardinal	-	-	-	MajorCurrency	-	<MajorCurrency> 2	
hundred	MagnitudePop1	-	-	No	-	-	0	
five	Cardinal	-	-	No	-	-	5	
dollars	CurrencySymbol	-	No	-	-	MajorCurrency	$ <\MajorCurrency>	$205

Table 3: *Excerpts from the rewrite table.*

Option	Spoken Form	Written Form
	...	
Cardinal	one	1
Cardinal	two	2
	...	
Magnitude	hundred	00
MagnitudePop1	hundred	0
	...	
MagnitudePop1	million	000,00
MagnitudePop2	million	000,0
	...	
AbbreviateMeasure	pounds	lbs.
AbbreviateMeasure	zettabytes	ZB
	...	
CurrencySymbol	pounds	£
	...	

Table 4: *The append table.*

Option	String
Period	.
Comma	,
Colon	:
Hyphen	-
Slash	/
Square	²
Cube	³

we choose the label sequence given the highest score by a label sequence bi-gram. The label sequence bi-gram is trained on a subset of the training utterances where $S_{labeled}$ had only one path.

3. Modeling

3.1. Structure

Given the task formulation, the modeling problem is reduced to labeling a sequence of tokens. To this problem, we apply a bi-directional LSTM [16, 17]. The input to the LSTM for a spoken-form token is computed by summing a linearly-projected token embedding with linearly-projected feature embeddings for all active features. The output of the LSTM is fed into a multi-layer perceptron (MLP) followed by a softmax layer with an output target for each label.

3.2. Features

Input features are binary and of two types: token lexical features and features derived from the rewrite table. We have a single lexical feature, which indicates whether the first letter of the token string is capitalized. This helps to determine when the *Capitalize* rewrite needs to apply. It also allows information to be propagated from the LM about, for example, which tokens are likely proper nouns. Note that the value for this feature is determined at run time, and thus it is informative even for words not in the input vocabulary.

Features derived from the rewrite table are based on which rewrite options can apply to a token. For example, looking at Table 3, "pounds" and "zettabytes" will both have an active feature corresponding to the *AbbreviateMeasure* option. "pounds" will also have an active feature corresponding to the *CurrencySymbol* option. These features are intended to help the model to predict the correct label even for infrequent tokens.

4. Experiments and Results

4.1. Data and Evaluation Criteria

We will present results using data from an English virtual assistant application. In lieu of having training and test data transcribed, we use an existing rule-based system to generate written-form output from spoken-form data. By manually checking 1000 randomly selected outputs, we estimate the sentence accuracy (SA) of the rule-based system at about 99%. Given the high accuracy of the rule-based system, we believe that using this data provides a reasonable way to evaluate our approach. We expect our results to generalize to the case where hand-transcribed data is available.

We will primarily use sentence fidelity (SF) to the rule-based system as our evaluation metric. A written-form output must perfectly match the rule-based system's output to be counted as matching. We present results on a set of 800,000 randomly selected utterances.

4.2. Data Selection

For about 80% of utterances in our data set, the only action performed by ITN is to capitalize the first spoken-form token. For this reason, we found it useful to over-represent utterances where ITN was doing a more significant transformation in training. We do this by selecting half of the training data using a regular expression matched against written-form utterances. This regular expression matched digits, commas, address abbreviations (e.g., "Ave."), currency symbols, common URL prefixes, common URL suffixes and the at symbol ("@").

4.3. Hyperparameters

After a coarse tuning, the most compact model on which we observed the best performance with 1M training utterances had 64 dimensional word embeddings, 32 dimensional feature embeddings, one bi-directional LSTM layer with 96 dimensional input

and 128 hidden units, and a single-layer MLP with 128 units. All weights and embeddings were randomly initialized using a uniform distribution over (-0.1, 0.1). The input spoken-form token vocabulary size was 30,000. All unique labels found in the training set were included in the output vocabulary. Because some labels occur very rarely, the output label vocabulary size depended on the training set. The output vocabulary sizes were 152, 170 and 208 for the 500K, 1M and 5M utterance training sets, respectively.

4.4. Results

In Table 5, we show the SFs achieved without input features, using only the capitalization feature and using all input features for various amounts of training data. We observe that using the capitalization feature gives significant gains across training data amounts. Looking closely at the results reveals that the most significant impact of this feature is rather mundane: it allows the model to correctly predict whether or not the first spoken-form token in an utterance needs to be capitalized, even when that token is very infrequent in the training data or out-of-vocabulary.

Adding features derived from the rewrite table increases accuracy slightly when training with 500K training utterances but does not appear to affect performance for larger training data amounts. However, it is easy to construct test examples for which the additional features prove useful. For instance, models built with only the capitalization feature will not correctly abbreviate measurement units not seen in the training data (e.g. "zettabytes".) Models trained with the additional input features will usually abbreviate these units correctly. Units not seen in the training data are extremely rare in the test data, and so we do not see an impact on overall accuracy, but we consider the consistent abbreviation of units a requirement for the system.

Table 5: *SF for models trained without input features, using only the capitalization feature, and using all input features using 500K, 1M and 5M utterances.*

Features	SF(%)		
	500K	1M	5M
None	98.95	99.26	99.65
Capitalization	99.33	99.62	99.85
All	99.46	99.62	99.85

To get a better handle on performance for those utterances where ITN performs significant transformations, we extracted targeted subsets of the test data. The sets were selected using regular expressions applied to utterance spoken forms. Examples are given for each in Table 6. To show that our reference rule-based system performs well on these sets, we also provide a rough estimate of its accuracy, based on manually evaluating 100 utterances per set. In Table 7, we present results using 500K, 1M and 5M training utterances. We see that with 1M training utterances, SF on all of the sets but CURRENCY exceeds 90%. We suspect that a more sophisticated training data selection approach would narrow the performance gap between training with 1M and 5M utterances.

To evaluate whether the system would respond as expected to training data modification, as well as whether it could capture transformations that occur in varied contexts, we attempted to improve the performance of the system on time expressions. The rule-based system requires the presence of 'a.m.' or 'p.m.' or a small set of trigger phrases in order to format a numeric expression as a time. This results in improperly formatted output like "Change the 5 PM alarm to 512". To correct these sorts of errors, we selected utterances with likely errors from the training data using a broad regular expression, then manually corrected the written forms. We did the same for the test set, obtaining a set of 400 utterances to evaluate the effect of the fix. After retraining the system on the edited training data, it produced the correct output for 84.8% of the test utterances. In general, we've had success at fixing a variety of incorrect behaviors by correcting training data as just described, adding targeted training data and, occasionally, adding new input features.

Table 6: *Targeted test set sizes, rule-based reference system SAs and examples. SAs were estimated on 100 utterances per set.*

Set	# Utts	SA(%)	Spoken Form Examples
TIME	23066	98	set an alarm for ten thirty a.m.
MATH	10954	97	what is thirty five times three
ADDRESS	7325	90	navigate to one fifty South Avenue Presto New Jersey
MEASURE	2525	99	how many pounds is twenty eight grams
DATE	1744	100	what day is January thirtieth two thousand seventeen
CURRENCY	825	95	how many dollars is thirteen hundred euros

Table 7: *SF on targeted test sets using models trained with 500K, 1M and 5M utterances.*

Set	SF(%)		
	500K	1M	5M
ALL	99.5	99.6	99.9
TIME	99.1	99.2	99.7
MATH	92.8	94.6	97.9
ADDRESS	86.7	91.9	97.7
MEASURE	85.6	96.5	98.5
DATE	88.0	92.1	98.1
CURRENCY	84.0	89.5	95.6

5. Conclusions

We present an approach to ITN that combines a set of simple rules and a few hand-written grammars with a bi-directional LSTM. The approach has the benefits of a data-driven system, while using an underlying statistical model that is relatively simple and compact. Using 1M training utterances, we are able to build a system which approaches the overall performance of a rule-based system with about 99% SA. We show that the approach also performs well on test sets targeted toward utterances where ITN performs significant transformations.

6. Acknowledgments

We thank Ladan Golipur for useful discussions on her work on TN that shaped important aspects of the approach described here. We also thank Yun Tang and Rongqing Huang for their assistance with the neural network toolkits used in this work.

7. References

[1] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer speech & language*, vol. 15, no. 3, pp. 287–333, 2001.

[2] Y.-C. Ju and J. Odell, "A language-modeling approach to inverse text normalization and data cleanup for multimodal voice search applications." in *Proc. INTERSPEECH*, 2008, pp. 2179–2182.

[3] M. Shugrina, "Formatting Time-Aligned ASR Transcripts for Readability," in *Proc. HLT*, 2010, pp. 198–206.

[4] H. Sak, Y.-h. Sung, F. Beaufays, and C. Allauzen, "Written-domain language modeling for automatic speech recognition," in *Proc. INTERSPEECH*, 2013, pp. 675–679.

[5] T. Schlippe, C. Zhu, J. Gebhardt, and T. Schultz, "Text normalization based on statistical machine translation and internet user support." in *Proc. INTERSPEECH*, 2010, pp. 1816–1819.

[6] R. Sproat and N. Jaitly, "RNN Approaches to Text Normalization: A Challenge," *arXiv preprint arXiv:1611.00068*, 2016.

[7] G. Neubig, Y. Akita, S. Mori, and T. Kawahara, "Improved statistical models for smt-based speaking style transformation," in *Proc. ICASSP*, 2010, pp. 5206–5209.

[8] A. Aw, M. Zhang, J. Xiao, and J. Su, "A phrase-based statistical model for SMS text normalization," in *Proc. ACL*, 2006, pp. 33–40.

[9] O. Divay, M. Bisani, P. Vozila, and J. Adams, "Automatic editing in a back-end speech-to-text system," in *Proc. HLT*, 2008, pp. 114–120.

[10] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Part-of-speech tagging with bidirectional long short-term memory recurrent neural network," *arXiv preprint arXiv:1510.06168*, 2015.

[11] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," *arXiv preprint arXiv:1508.02096*, 2015.

[12] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," *arXiv preprint arXiv:1604.05529*, 2016.

[13] M. Lewis, K. Lee, and L. Zettlemoyer, "LSTM CCG Parsing," in *Proc. HLT*, 2016, pp. 221–231.

[14] A. Vaswani, Y. Bisk, K. Sagae, and R. Musa, "Supertagging With LSTMs," in *Proc. HLT*, 2016, pp. 232–237.

[15] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The opengrm open-source finite-state grammar software libraries," in *Proc. ACL*, July 2012, pp. 61–66.

[16] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.