MINIMALLY SUPERVISED WRITTEN-TO-SPOKEN TEXT NORMALIZATION

Axel H. Ng, Kyle Gorman, and Richard Sproat

Google, Inc.
111 8th Ave., New York, NY, USA

ABSTRACT

Text normalization is the task of converting from a *written* representation into a representation of how the text is to be *spoken*. For most real-world speech applications, the text normalization engine is developed mostly by hand. For example, a hand-built grammar may be used to enumerate possible ways to say a given token in a given language, and a statistical model used to select the most appropriate verbalizations in context. We examine the tradeoffs associated with using more or less language-specific knowledge for text normalization. In the most data-rich scenario, we have access to a carefully constructed hand-built normalization grammar that for any given token will produce a lattice of all possible verbalizations for that token. We assume a parallel corpus of aligned written-spoken utterances. As a substitute for the hand-built grammar, we consider a language-universal normalization covering grammar, where the developer merely needs to provide a set of lexical items particular to the language. As a substitute for the aligned corpus, we consider a scenario where one only has the spoken side, and the corresponding written side is "hallucinated" by composing the spoken side with the inverted normalization grammar. We report performance of the above scenarios on experiments with English and Russian.

Index Terms— speech synthesis, speech recognition, natural language processing.

1. INTRODUCTION

Over the past 15 years there has been substantial progress applying machine-learning approaches to text normalization [1, 2, 3, 4, 5, 6]. Nonetheless, for real-world speech applications, such as text-to-speech synthesis (TTS) or automatic speech recognition (ASR), text normalization engines require a substantial amount of manual grammar development. One reason for this is a lack of appropriate annotated data for training models. Constructing hand-built grammars is a labor intensive and time consuming process, but so is constructing appropriate corpora consisting of raw text and the corresponding normalization.[1] Training corpora for text normalization must be constructed for the particular application they are intended for. While corpora exist for social media text normalization [3, 4, 6], these are of little use for a TTS system, since such systems rarely consume examples like *cu l8tr lv u ;-)*. On the other hand, TTS systems must expand numbers, times, currency amounts, and the like into spoken words: e.g., for an input *Fully grown giraffes stand 4.3–5.7 m (14.1–18.7 ft) tall*, one might expect text normalization to produce *Fully grown giraffes stand four point three to five point seven meters (fourteen point one to eighteen point seven feet) tall*. Such transductions are of little import for social media text normalization.

In this paper we present a collection of methods to learn a mapping from *written* text to its *spoken* form, i.e., the mapping needed for a TTS system, or for producing ASR language model training data from raw text.[2] Some recent work [8] has framed this as a supervised sequence-to-sequence problem, but results there suggest that this unconstrained approach may require millions or tens of millions of labeled tokens. We are primarily interested in developing speech-oriented text normalization systems for new languages, and in particular, low-resource languages for which we have access to text data and some linguistic expertise, but lack the resources for large-scale grammar development or manual labeling,[3] and thus we focus on minimally supervised approaches.

We make use of 2.24 million words of English transcribed speech and 1.95 million words of Russian speech, both collected from a voice search application. This *spoken* data was automatically converted to plausible *written* forms, then the written forms were corrected by native-speaker annotators, once personally identifiable information was removed. This results in 1.75 million written tokens for English, and 1.59 million written tokens for Russian; In all experiments, we depend on a grammar which defines the hypothesis space of possible spoken forms for a given written sentence. These

[1] Minimally supervised approaches using *unannotated* text data based on matching unnormalized forms with possible expansions given some corpus of 'clean' text [1, 6, 7] are only applicable to certain classes of tokens, such as abbreviations.

[2] Here we ignore pronunciation-oriented tasks such as homograph disambiguation or grapheme-to-phoneme conversion.

[3] One may ask whether it would be possible to collect parallel data in some high resource language and then automatically translate both written and spoken sides into the target language. Unfortunately, current machine translation systems are particularly poor at handling categories like numbers and currency amounts. For example, Google's English-to-Korean system translates *nineteen eighty four* as *yeol yeodeolp* 'ten eight', despite the fact that this is a relatively common expression (due to Orwell's famous novel).

grammars are compiled into weighted finite-state transducers (WFSTs) using the Thrax grammar development library [9]. For example, for the cardinal number *123*, this grammar might produce *one hundred twenty three*, *one two three*, *one twenty three*, etc. In Section 3 we describe how the grammars are combined with a reranker to select the best hypothesis given the local input context (i.e., nearby written words).

In some experiments we use hand-built *language-specific normalization grammars*, and specifically, pre-existing grammars developed as part of a system for training language models for ASR. This grammar is designed to *overgenerate*; that is, it produces, for any input token, a (potentially weighted) lattice of possible spoken strings. These language-specific grammars are written by native speaker linguists and designed to cover a range of types of normalization phenomena, including numbers, some mathematical expressions, some abbreviations, times, and currency amounts. We refer to these various categories in need of normalization as *semiotic classes* [10].

Developing these grammars for a new language requires significant effort and knowledge, but there is a lot of structure across shared across languages, and this can be exploited if the grammars are appropriately parameterized. For instance, when reading a time expression like *3:30 PM*, we might first ask whether the "period" *PM* is read after, as in English *three thirty PM*, or before, as in Chinese, Japanese, or Korean? It is thus possible to construct a *covering grammar*, which simply generates all options known to be present in any language under consideration. All that is required to specialize the covering grammar for a specific language is to provide a small lexicon providing verbalizations for individual terms (e.g., how *PM* is read): such a lexicon could be as small as a few hundred items, and does not require much linguistic sophistication to produce. While the effort required varies greatly by language, we estimate that developing language-specific normalization grammar requires roughly a week, whereas lexicalizing a covering grammar (and if necessary, expanding it to handle patterns not yet seen) for a specific language can be done in a day or less. Something more than this is required for readings of numbers, which are considerably less constrained: for that we make use of the algorithm described in [11], which can learn verbalizations for number names in a language from about 300 labeled examples. In Section 3, we describe the use of a covering grammar in place of the more-carefully engineered language-specific grammar, and measure the degradation resulting from replacing the language-specific grammar with the more-permissive covering grammar, which also has to decide whether one reads the equivalent of *PM* before or after *3:30*. Thus we have two kinds of normalizers: one derived from a hand-constructed language-specific grammar, and the other from a permissive language-universal covering grammar composed with a language-specific lexicon.

Finally, we consider a low-resource scenario where we have only the *spoken* data, but do not the corresponding written forms. This may obtain if we have access to speech transcriptions, or a corpus of fairly clean written text where numbers, times, etc., are verbalized. Our approach in this scenario is to "hallucinate" the written side by inverting the normalizer to turn it into a *denormalizer* [12, 13], and then composing the resulting lattice of putative written forms with the normalizer; we describe this process in Section 3. This final scenario has much in common with prior work by [14] and [15]. These scenarios yield four training conditions:

1. language-specific grammar with real data
2. language-specific grammar with hallucinated data
3. covering grammar with real data
4. covering grammar with hallucinated data

As one might expect, the first condition usually yields the best results and, the fourth condition the worst results. But we are most interested in just how much accuracy is lost under lower-resource conditions, and we report in detail on this below.

	Sents.	Written tokens	Spoken tokens
English			
Train.	325K	1.75M	2.24M
Devel.	5,000	27,602	34,899
Test.	5,000	28,225	35,645
Russian			
Train.	348K	1.59M	1.95M
Devel.	2,000	9,291	11,305
Test.	2,000	8,874	11,002

Table 1. Corpus size statistics for English and Russian.

2. DATA AND CONVENTIONS

As noted above, our data consists of English and Russian speech from a voice search application, transcribed in both written and spoken forms. Each sentence contains at least one token that requires normalization. Disjoint subsets are held out from the corpora as development and test sets. Table 1 lists basic statistics for the English and Russian corpora. In English, roughly 62% of the tokens are passed through unchanged by normalization; the figure is 59% for Russian.

The parallel data is aligned at the word level, so that each written token aligns to one or more spoken tokens. We represent a training sentence as a sequence of pairs $(x_1, \mathbf{z}_1), \ldots, (x_n, \mathbf{z}_n)$, where x_i is a written token and $\mathbf{z}_i$ is the sequence of spoken tokens it aligns to. Note that here and below, we use lowercase letters (e.g., x) to represent words and scalars and bold lowercase letters (e.g., $\mathbf{y}$) for sequences of words and for vectors. For any operation that takes a string operand (e.g., composition), we consider a word sequence

x_1 = wake $\mathbf{z}_1$ = wake
x_2 = me $\mathbf{z}_2$ = me
x_3 = at $\mathbf{z}_3$ = at
x_4 = 9:00 $\mathbf{z}_4$ = nine
x_5 = am $\mathbf{z}_{5,6}$ = a m

Fig. 1. An example word-aligned sentence, with a written-side input and the spoken (verbalized) output.

$\mathbf{y}$ to be synonymous to the string constructed by joining the elements of $\mathbf{y}$ with space. An example is shown in figure 1.

As discussed above, we use two types of grammar: language-specific normalization grammars and covering grammars. The language-specific grammar for English consists of about 2,500 lines of rules and lexical specifications written using the Thrax toolkit [9], and the equivalent Russian grammar has about 4,100 lines. The language-independent covering grammar consists of about 700 lines of Thrax. In addition we have 75 lexical specifications for English and 220 for Russian, with the bulk of the difference being due to the need to list various potential inflected forms in Russian. As noted earlier, number grammars are specified separately using the method described in [11]. The covering grammar is intended to represent all possible ways in which a language might express particular semiotic classes. By way of illustration, consider the following Thrax grammar fragment:

```
period = @@TIME_AM@@ | @@TIME_PM@@;
space = " " | ("" : " ");
time = Optimize[(period space)? time_variants |
                time_variants (space period)?];
```

The first expression defines `period` to be either `@@TIME_AM@@` or `@@TIME_PM@@`, which will be verbalized with appropriate lexical items in the target language. The third expression then defines a time to be any of a number of ways to verbalize core time values—e.g., *three thirty* or *half past three*, termed `time_variants` here—either preceded or followed by an optional `period` and an intervening space. For any target language, this will *overgenerate*, so we require a means of selecting the appropriate form for the language; see Section 3. The lexical map for English indicates that `@@NOVEMBER@@` can be verbalized as `november`, `@@DECIMAL_DOT@@` as `point`, and `@@URL_DOT@@` as `dot`. Clearly there is more variation across languages than is captured in the above expression: for example some languages habitually make a finer-grained set of distinctions than just *AM* vs. *PM*. Russian, for example, distinguishes between *in the evening* and *at night*. (English obviously can also make such distinctions, but *AM* and *PM* are sufficient in most cases.) However, it is not the case that one must provide a new grammar for each new language. For a number of languages—Chinese, Japanese, and Korean, among others—it is common to specify a time as before or after noon. As with number names [11], one can parameterize the grammar to cover a wide range with ways in which languages express times and other semiotic classes.

In developing the covering grammar we endeavored to cover the same phenomena as covered in the language-specific verbalization grammars. For example the English verbalization grammar handles various ways of reading digit sequences, so that *990* might be read as *nine hundred ninety*, or *nine ninety* or *nine nine oh*, all of which are possible in various contexts. In a similar fashion we also allowed digit sequences in the covering grammars to be read as cardinal (or ordinal numbers), or as various combinations of these two categories. Obviously a fair amount of genre-specific knowledge goes into the design of the covering grammar, and without this knowledge, this strategy would fail. On the other hand, if care is taken in the construction of the grammar to allow for the different ways in which languages might verbalize various semiotic classes, then developing a system for a new language merely requires one to specify a lexical map, which requires far less work and expertise than developing a new grammar from scratch. As noted above, some languages will require extensions to the covering grammar (e.g., finer distinctions than simply *AM* and *PM*), but this is still less effort than developing a complete normalization grammar.[4]

3. RANKING MODELS

This section applies equally to the language-specific normalizer grammar and the covering grammar, and so in what follows we refer to both as simply *the grammar*.

For a written token x_i in sentence $\mathbf{x}$, the set of possible outputs Y_i consists of both the result of composition of x_i with the grammar as well as x_i itself (which corresponds to passing the token through, unmodified). When $|Y_i| > 1$—i.e., when there are multiple normalization options for x_i—we need some way to choose the contextually appropriate output $\mathbf{y}_i^* \in Y_i$ which is closest to the reference verbalization $\mathbf{z}_i$. The simplest such model is an n-gram language model (LM) built from the spoken side of the parallel training data [1]. We refer to this as the *baseline system*. However, there are two limitations with this approach. First, most of the LM's parameters are otiose, as they refer to words passed through during normalization. Secondly, this does not use any features of the written input.

We therefore cast the choice of $\mathbf{y}_{i,j}$ as a ranking problem. For each Y_i, let $G_i = \{\mathbf{y} \mid d(\mathbf{y}, \mathbf{z}_i) = \min_{\mathbf{y}'} d(\mathbf{y}', \mathbf{z}_i)\}$ be the subset of *good* candidates from Y_i, according to some distance metric $d(\cdot)$ such as label edit distance. Given a feature function $\Phi(\mathbf{y}_{i,j}, \ldots)$ combining the candidate verbalization and properties of the context into a feature vector, we train

[4] In the hopes that they will be more widely useful, we have released the covering grammars and lexical maps for English and Russian under the Apache 2.0 license: `http://github.com/google/TextNormalizationCoveringGrammars`

a maximum entropy ranker [16] by choosing $\mathbf{w}$ to maximize

$$L(\mathbf{w}) = \sum_i \log \frac{\sum_{\mathbf{y}_{i,j} \in G_i} \exp\langle \mathbf{w}, \Phi(\mathbf{y}_{i,j}, \ldots)\rangle}{\sum_{\mathbf{y}_{i,j} \in Y_i} \exp\langle \mathbf{w}, \Phi(\mathbf{y}_{i,j}, \ldots)\rangle}$$

Training and inference algorithms depend on the feature function. We consider two classes of feature functions, as follows.

3.1. Local ranking

We can use a local feature function $\Phi_I(\mathbf{X}, i, \mathbf{y})$, which sees the whole input and the current verbalization, but not the verbalization of any other written tokens. Such feature functions allow independent inference for each written token. For training, we simply generate one training example for each x_i for which $|Y_i| > 1$. We use the following features:

Local output n-grams: n-grams ($1 \leq n \leq 3$) in output y_i;

Boundary trigrams: two written words on the left (x_{i-2}, x_{i-1}) and the first word of y_i; two written words on the right (x_{i+1}, x_{i+2}) and the last word of y_i;

Written/spoken skip-grams: pairs of one written word on either the left or the right within a 4-word window and an output word in y_i;

Bias: $1_{x_i = y_i}$, i.e., whether x_i is passed through.

3.2. Discriminative language model

In normalization, the majority of the written tokens are actually passed through. Therefore a feature function

$$\Phi_O(\mathbf{y}_{1,j_1}, \ldots, \mathbf{y}_{i,j_i}, 1_{x_i = \mathbf{y}_{i,j_i}})$$

that sees the verbalization history but not the written tokens actually has a lot of overlapping information with $\Phi_I(\cdot)$. We therefore limit the features to spoken token n-gram suffixes ending at $\mathbf{y}_{i,j}$ and the bias $1_{x_i = \mathbf{y}_{i,j}}$. Further, we fix the weight of the bias feature to a constant negative number because passing through should be discouraged, leaving n-gram weights as the only tunable parameters. This trained model can be encoded as a WFST for efficient inference [17].

This feature parameterization also allows us to train a model from spoken tokens with no information about the written tokens: we "hallucinate" the written side. Consider the following simple training procedure with $\Phi_O(\cdot)$: for each Y_i, we extract feature vectors by assuming all the previous verbalizations are correct, i.e., $\Phi_O(\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{y}_{i,j}, 1_{x_i = \mathbf{y}_{i,j}})$, and train $\mathbf{w}$ to assign higher scores to those $\mathbf{y}_{i,j} \in G_i$. If we only have spoken tokens $\mathbf{Y}$ but not the corresponding written tokens, we can approximate the above example for any sub-sequence $\mathbf{y}_i$ in $\mathbf{Y}$ by approximating Y_i with $Y'_i = \pi_o(\mathbf{y}_i \circ V^{-1} \circ V)$, where V is the WFST representing the grammar, and G_i by $G'_i = \{\mathbf{y}_i\}$. The output of the composition $\mathbf{y}_i \circ V^{-1}$ is the set of written tokens from which the grammar may produce $\mathbf{y}_i$. Then, the cascaded composition gives all the spoken tokens the grammar can produce given a written token that produces $\mathbf{y}_i$. If V has perfect coverage over the written tokens, then Y'_i is a superset of Y_i and G'_i is a subset of G_i. To control the amount data generated, we only allow $\mathbf{y}_i$ to be at most 5 words, and limit the size of Y'_i to be the 10,000 shortest paths through the cascade. For example, the spoken sequence *one twenty* might be mapped via V^{-1} to various written forms such as *120, 1:20, one20*…. When composed with V, these yield potential spoken forms, including *one twenty, one hundred twenty, twenty past one*….

3.3. Number grammar biasing

Although the set of normalization candidates for a single written token almost never exceeds 100 for English, the number of candidates in Russian can be prohibitively large. This is because the Russian number covering grammars permits all possible morphological variants for each output word, leading to combinatorial explosion. We therefore bias the output by composing the number grammar component of the Russian covering grammar with a language model over the spoken (output) side of the number grammar. LM training data is mined by matching web text to the output projection of the unbiased grammar [18].

3.4. Discriminative LM vs. local ranking

In preliminary experiments, we observed that the discriminative LM trained on real data underperformed the local ranking model, particularly with language-specific covering grammars. This is because two pieces of data are not available with hallucinated data, the most interesting use case for the discriminative LM technique:

Tuned pass-through bias: When training on real data, the bias is tuned along with the n-gram weights, but it cannot be tuned using hallucinated data. Therefore we fix the bias to optimize performance on the development set.

Spoken phrase boundary: Consider the input *1911 9mm*, verbalized as *nineteen eleven nine millimeter*. The discriminative LM prefers *one nine one one nine millimeter*, because the hallucinated written form includes *19119 mm*, and there is a strong data bias to read 5-digit numbers as digit sequences. With real data, we know where the spoken phrase for a single written word begins or ends, and can train discriminative LMs on data with a marker <*p*> inserted at such boundaries. Thus the LM can distinguish between *one nine one one* <*p*> *nine* and *one nine one one nine*, and note that the former is less likely than *nineteen eleven* <*p*> *nine*.

4. RESULTS

We evaluate the systems using the following two metrics:

	Language-specific		Covering	
	WER	SER	WER	SER
English				
Baseline	10.02	14.14	13.65	20.06
Local ranking	7.52	10.38	10.14	13.40
DLM	8.59	11.90	10.57	14.06
DLM (TB)	8.14	11.18	10.36	13.82
DLM (TB, SPB)	7.50	10.38	10.49	13.82
DLM (H data)	8.80	12.06	10.92	14.52
Russian				
Baseline	22.29	25.45	26.05	35.25
Local ranking	14.85	19.85	17.66	24.50
DLM	18.79	23.75	17.79	23.45
DLM (TB)	15.54	20.25	17.70	23.35
DLM (TB, SPB)	15.28	19.90	17.47	23.10
DLM (H data)	18.44	22.30	19.67	26.05

Table 2. Error rates for English and Russian.

Word Error Rate (WER): The minimal number of word edits between predicted and reference outputs, divided by the number of tokens in the reference output.

Sentence Error Rate (SER): The proportion of test sentences with at least one error in the output. For the baseline systems, we use unpruned trigram language models trained on the spoken side of the training data, with Katz backoff; experiments using higher-order n-gram models did not lower WER on the development set. We experiment with the variants of discriminative LM (DLM) discussed in Section 3.4, adding *tuned bias* (TB) and *spoken phrase boundary* (SPB) when training on real data. Finally, we experiment with hallucinated data (H data) for training the discriminative LM. For all systems, we set hyperparameters so as to minimize WER on the development set.

Results for English and Russian are shown in Table 2. Ranking is an improvement over the baseline in all cases. Indeed in Russian, all ranking-based systems outperform both baselines regardless of the grammar used. The cost of using the covering grammar as opposed to the language-specific grammar is about a 25% relative WER for English, but only about 6% relative for Russian, suggesting that the Russian (language-specific) grammar is less constrained, or simply less carefully curated, than the English grammar.

Local ranking with all the features discussed in Section 3.1 yields the best results, with a penalty for the discriminative LM (Section 3.2) with a language-specific grammar of between 14% relative (English) to 25% relative (Russian) WER; this is presumably due to the former method's richer feature set. The discriminative language model ranking approaches the performance of local ranking once tuned bias and phrase boundary information are available. Finally, the penalty associated with hallucinated data is not large. The discriminative LM and local ranking are also much closer to each other in performance when using a covering grammar, and in Russian, the best discriminative LM outperforms local ranking.

As expected, error rates are higher for Russian. This presumably reflects the greater morphological complexity of this language and the aforementioned possibility that the Russian language-specific grammar is inferior to the English grammar.

Take together, the results suggest that one can develop an initial system with reasonable performance for a new language with very little work if one just has a source of "spoken" text, and is willing to invest a small amount of effort to produce a list of lexical entries and some example number names [11]. Naturally, parallel data and language-specific grammar development improves performance.

5. QUALITATIVE ERROR ANALYSIS

We performed a qualitative error analysis in order to see if there were any broad generalizations about types of errors found in the different training scenarios.

5.1. English

Common classes of English errors include:

- The covering grammar does not generate appropriate readings for certain numerical expressions (e.g., *49ers*).
- The covering grammar does not generate the "hundreds" readings of numbers (e.g., reading *2200* as *two thousand two hundred* but never *twenty two hundred*).
- Covering grammar rankers occasionally under- or overgenerate letter or digit sequences (e.g., reading *something* as the letter sequence *s o m e t h i n g*).
- Hallucinated data rankers prefer to read + as *and* even when the context requires it to be read as *plus*.

5.2. Russian

Common classes of Russian errors include:

- The covering grammar does not generate the *dot* reading for . in URLs.
- The covering grammar does not generate *раз*, the special 'counting' form of *1*.
- Covering grammar rankers occasionally use cardinal numbers where ordinals are required, e.g., in dates.
- Hallucinated data rankers occasionally under- or overgenerate letter or digit sequences (e.g., reading *AK-47* as the equivalent of *ak forty seven*).

6. CONCLUSIONS

We have explored the relative contribution of different language resources to the performance of text-normalization systems trained using ranking. Specifically:

- language-specific normalizer grammars versus language-independent covering grammars with a small amount of language-specific lexical knowledge; and
- aligned written-spoken data versus only spoken data "hallucinating" the written side.

We find that the performance degradation for using a covering grammar over a language-specific grammar need not be large. In a similar fashion, the degradation caused by hallucinated data need also not be large.

The choice of English and Russian in these experiments was motivated by the fact that we already had text normalization systems for these languages and could thus compare directly with existing approaches. But our real interest is in developing systems for new languages, and in particular, low-resource languages for which we have access to raw data and some linguistic expertise, but lack the resources for large-scale grammar development or manual labeling. We intend to apply these methods to low-resource languages in future work.

7. REFERENCES

[1] Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards, "Normalization of non-standard words," *Computer Speech and Language*, vol. 15, no. 3, pp. 287–333, 2001.

[2] Sarah Schwarm and Mari Ostendorf, "Text normalization with varied data sources for conversational speech language modeling," in *ICASSP*, 2002, pp. 789–792.

[3] Bo Han and Timothy Baldwin, "Lexical normalization of short text messages: Makn sens a #twitter," in *ACL*, 2011, pp. 368–378.

[4] Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu, "Insertion, deletion, or substitution? Normalizing text messages without pre-categorization nor supervision," in *NAACL*, 2011, pp. 71–76.

[5] Deana Pennell and Yang Liu, "Toward text message normalization: Modeling abbreviation generation," in *ICASSP*, 2011, pp. 5364–5367.

[6] Yi Yang and Jacob Eisenstein, "A log-linear model for unsupervised text normalization," in *EMNLP*, 2013, pp. 61–72.

[7] Brian Roark and Richard Sproat, "Hippocratic abbreviation expansion," in *ACL*, 2014, pp. 364–369.

[8] Richard Sproat and Navdeep Jaitly, "An RNN model of text normalization," in *INTERSPEECH*, 2017.

[9] Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai, "The OpenGrm open-source finite-state grammar software libraries," in *ACL*, 2012, pp. 61–66.

[10] Paul Taylor, *Text to speech synthesis*, Cambridge University Press, Cambridge, 2009.

[11] Kyle Gorman and Richard Sproat, "Minimally supervised number normalization," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 507–519, 2016.

[12] Maria Shugrina, "Formatting time-aligned ASR transcripts for readability," in *NAACL*, 2010, pp. 198–206.

[13] Lucy Vasserman, Vlad Schogol, and Keith Hall, "Sequence-based class tagging for robust transcription in ASR," in *INTERSPEECH*, 2015, pp. 473–477.

[14] K. Sagae, M. Lehr, E. Prud'hommeaux, P. Xu, N. Glenn, D. Karakosc, S. Khudanpur, B. Roark, M. Saraçlar, I. Shafran, D. Bikel, C. Callison-Burch, Y. Cao, K. Hall, E. Hasler, P. Koehn, A. Lopez, M. Post, and D. Riley, "Hallucinated n-best lists for discriminative language modeling," in *ICASSP*, 2012, pp. 5001–5004.

[15] Noah Smith and Jason Eisner, "Contrastive estimation: Training log-linear models on unlabeled data," in *ACL*, 2005, pp. 354–362.

[16] Richard Sproat and Keith B. Hall, "Applications of maximum entropy rankers to problems in spoken language processing.," in *INTERSPEECH*, 2014, pp. 761–764.

[17] Ke Wu, Cyril Allauzen, Keith B. Hall, Michael Riley, and Brian Roark, "Encoding linear models as weighted finite-state transducers," in *INTERSPEECH*, 2014, pp. 1258–1262.

[18] Richard Sproat, "Lightly supervised learning of text normalization: Russian number names," in *IEEE Workshop on Speech and Language Technology*, 2010, pp. 436–441.