Improving homograph disambiguation with supervised machine learning

Kyle Gorman, Gleb Mazovetskiy, Vitaly Nikolaev
Google Inc.
{kbg,glebm,vitalyn}@google.com

Abstract

We describe a pre-existing rule-based homograph disambiguation system used for text-to-speech synthesis at Google, and compare it to a novel system which performs disambiguation using classifiers trained on a small amount of labeled data. An evaluation of these systems, using a new, freely available English data set, finds that hybrid systems (making use of both rules and machine learning) are significantly more accurate than either hand-written rules or machine learning alone. The evaluation also finds minimal performance degradation when the hybrid system is configured to run on limited-resource mobile devices rather than on production servers. The two best systems described here are used for homograph disambiguation on all US English text-to-speech traffic at Google.

Keywords: Homograph disambiguation, machine learning, text normalization, text-to-speech synthesis

1. Introduction

Despite substantial progress in applying machine learning to text normalization and linguistic analysis for text-to-speech synthesis (TTS), it is still the case that most front-end processing in real-world TTS systems is done by language-specific hand-written rules. While rule systems provide an high degree of interpretability, they may require a great deal of development effort to obtain reasonable accuracy. Therefore, rule-based components represent substantial barriers for both quality control and internationalization.

1.1. Homograph disambiguation

When text input is sent to the Google TTS engine, it is first tokenized, and then pronunciations are selected for these tokens (Ebden and Sproat, 2014). For most tokens—such as in-vocabulary words with a single pronunciation—this requires only dictionary lookup. But other types of tokens, so-called *semiotic classes* (Taylor, 2009)—such as numbers, currencies and measures, dates and times, etc.—and out-of-vocabulary words, require additional language-specific processing (Sproat et al., 1992; Sproat et al., 2001). One particularly challenging class of tokens are *homographs*, polysemous words pronounced differently depending on the intended sense. One must analyze the context in which a homograph occurs to select a contextually appropriate pronunciation.

1.2. Sources of homography

Homography occurs any time two words pronounced distinctly are spelled the same. This may arise due to inflectional processes not indicated orthographically, as in the English irregular verb *read*, pronounced as either the past [ɹɛd] or the present [ɹiːd]. Alternatively, homographs may represent phonologically and semantically distinct lexical items which just happen to share a spelling, as in *bow*, pronounced either as [boʊ] or [baʊ]. We refer to these categories of homography as *morphosyntactic* and *lexical*, respectively.[1]

One may also discern a *mixed* category for those homographs which carry both morphosyntactic and semantic distinctions. For instance, *produce* may either be [pɹəˈdus], a verb referring to a process, or the noun [ˈpɹoʊdus], referring to the result of that process. At the same time, the verb has a number of semantic extensions not available for the noun; for instance, it may refer to overseeing the creation of musical recordings. Because the semantic relationship between the various pronunciations of a homograph are arbitrary, and because there may be hundreds or even thousands of homographs in a given language, homograph disambiguation requires substantial language-specific resources. Furthermore, it has been claimed that listeners' subjective evaluations of TTS are particularly sensitive to homograph disambiguation errors (Braga et al., 2007).

2. Rule-based disambiguation

Prior to the work described in this study, the Google TTS engine performed homograph disambiguation using language-specific rules curated by linguists and engineers. One major class of rules specifies the appropriate pronunciation to be used when the homograph occurs in the context of certain nearby words or phrases; for instance, one may specify that the homograph *used* is pronounced as [juːst] when immediately followed by the token *to*, as in the sentence "She *used* to smoke." Another class of rules selects a pronunciation to be applied when the homograph is tagged as a certain part of speech (POS). Finally, for each homograph one rule must be specified as a default, used when no more specific rules apply. Some limitations of such rule systems can be illustrated by considering a recent bug reported to us concerning the English lexical homograph *winds*. For this word, the nominal pronunciation [wɪndz] is the default, but rules selects the verbal pronunciation [waɪndz] when the homograph is tagged as a verb, or when it is immediately followed by the word *up*. Both the word context and the POS rule may seem sensible and intuitive, but both fail on a sentence like "There may be *winds* up to 20 miles per hour", for which the nominal pronunciation is required. Here, the word context rule will overapply, as will the POS rule if—as is usually the case in this context—the tagger incorrectly identifies *winds* as a verb. Furthermore, the choice of default rules is in large part based on introspection, and without empirical observations it is not clear, for example, which of the two pronun-

[1] This cuts across the traditional distinction between those homographs which can or cannot be disambiguated by part of speech (Yarowsky, 1997), since the distinct pronunciations of the latter category may or may not have overlapping parts of speech. For instance, the lexical homograph *console* is largely disambiguated by part of speech, but both pronunciations of the lexical homograph *bass*—[beɪs] and [bæs]—are usually nouns.

ciations of the English word *live*—[laɪv] or [lɪv]—ought to be considered the default. This issue is particularly important since default rules are the only rules yet available for many low-resource languages. Because of such limitations, prior to this work, homograph disambiguation was a major source of bug reports filed against the Google TTS front-end: it was the most buggy component for Russian, and the second most buggy one for English.

3. Machine learning-based disambiguation

We therefore consider the possibility of using machine learning methods to improve the existing rule-based system. Our basic design uses a set of multinomial classifiers—one per homograph—selecting the best resolution given a featural representation of the local context. Unlike prior work which uses complex and novel techniques to weigh conflicting sources of evidence and to prevent overfitting (Hearst, 1991; Sproat et al., 1992; Yarowsky, 1997; Silva et al., 2012), we simply use discriminative training and regularization.

3.1. Homograph disambiguation features

By representing this task as supervised classification, we are free to choose arbitrary features of the input context. We made use of the following set of features for both morphosyntactic and lexical homographs:

Word context features These represent tokens one or two to the left and/or right of the target homograph, left- and right-context bigrams, and a single skipgram centered on the target. For instance, consider the sentence "It was a terrific, riveting, really fast *read* and really exciting and really horrifying, but managed to be really touching." For this example the word context features are represented by the strings `WL2:really`, `WL1:fast`, `WR1:and`, `WR2:really`, `WL2:really_WL1:fast`, `WR1:and_WR2:really`, and `WL1:fast_WR1:and`. Following Yarowsky (1997), we use equivalence classes for context tokens classified as instances of specific *semiotic classes* such as numbers, currencies, measure expressions, or letter sequences; see Ebden and Sproat (2014) for a full list. Thus the featural representation of "1993 to *present*" and "2017 to *present*" are identical, since *1993* and *2017* are both instances of the `DATE` semiotic class.

POS tag feature Prior work on homograph disambiguation makes heavy use of features derived from part of speech (POS) tags or other automatic morphosyntactic analyses. We therefore process tokenized sentences with a POS tagger and extract a feature representing the hypothesized POS tag of the homograph itself.[2] However, this feature is only available when synthesis is performed on server; our embedded TTS engines (i.e., those running on mobile devices not connected to the internet) lack an on-device POS tagger.

Capitalization feature Following Sproat et al. (1992), we extract a feature indicating whether the target homograph is uppercase, titlecase, or lowercase. This feature is particularly useful for those homographs where one sense is a proper name and the other is not, as in *Polish* vs. *polish*.

[2] We also experimented with features derived from POS tags of nearby words, but this did not improve accuracy overall.

Type of ambiguity	Count	Example
Morphosyntactic	78	*read*: [ɹiːd] vs. [ɹɛd]
Lexical	62	*bow*: [boʊ] vs. [baʊ]
Mixed	23	*use*: [juːz] vs. [juːs]

Table 1: Counts and examples of three major categories of homographs in the data set.

3.2. Model training

Features are fed to a multinomial log-linear (i.e., maxent) model. This model is trained using an internal library employed for a number of other classification tasks in our TTS front-end, including sentence boundary detection and word stress prediction (Hall and Sproat, 2013; Sproat and Hall, 2014). During training, we use batched stochastic gradient descent with a fixed learning rate of $\alpha = .1$, and L_1 regularization. Separate models are trained for each homograph.

3.3. Model hybridization

While one could simply replace the rule-based system with machine-learned classifiers, we also consider a hybrid variant in which non-default rules pre-empt the learned classifiers, and learned classifiers pre-empt default rules. This approach is based on our intuition that non-default rules have high precision but low recall, and thus most errors are due to misapplication of default rules.

3.4. Outline

In what follows, we describe data collection and evaluation procedures, and compare the machine-learned model to the existing rule-based system using a manually labeled, publicly available database. We also estimate the performance degradation associated with embedded models which do not have access to a POS tagger.

4. Materials & methods

The following section describes data collection and evaluation methods used to compare systems.

4.1. Data collection

Querying an existing pronunciation lexicon for US English, we selected a set of 163 homographs for this experiment. Nearly all have two pronunciations, but two homographs have three pronunciations. To faciliate error analysis, one of the authors then coded each homograph as either *morphosyntactic*, *lexical*, or *mixed*; the counts of these three categories are given in Table 1. We then randomly sampled sentences containing these homographs from English-language Wikipedia articles, and manually filtered these to remove non-English text, bibliographic entries, and the like. This resulted in roughly 100 sentences per homograph. These examples were then labeled by a team of English-speaking annotators. Examples of the each homograph were grouped into batches of 20, and at the start of each batch, the annotator was provided with dictionary definitions for each pronunciation of a homograph as well as unique tags ("word IDs") for each. They were then presented with an example sentence and asked to select the best word ID for the homograph (marked with bold) in that sentence. Annotators

were also permitted to mark an example as "ambiguous", in which case the example was discarded.[3] Each example was labeled by three separate annotators; if all three did not assign the same word ID to an example, the disagreement was resolved by a fourth, more experienced annotator.

4.2. Data release

We have released this labeled data under an Apache 2.0 license.[4] The primary data consists of tab-separated values (TSV) files in which each row contains the identity of a homograph, its word ID label, the sentence itself, and byte-based indices for the exact location of the homograph token. For instance, a row might indicate that the homograph *read* occurs at bytes 41–45 in a certain given sentence and that it has the word ID `read_present` in this example. The data set is randomly split into two files, one containing a 10% random sample stratified by homograph, reserved for evaluation, and another contains the remaining data, intended for training and development. A supplementary TSV file can be used to match word IDs to an IPA transcription of the corresponding pronunciation; for example, this file indicates that `read_present` is pronounced [ɹiːd].

4.3. Evaluation

We evaluate models using two metrics: micro-accuracy and macro-accuracy (or *mean average accuracy*). The former is simply the percentage of examples correctly classified across all homographs; the latter is the arithmetic mean of the per-homograph accuracies. For the server model, the L_1 coefficient is tuned to maximize micro-accuracy on held-out data; for the embedded model it is also used to keep model size below a certain threshold for on-device data. [5]

5. Results

5.1. Baseline accuracies

Table 2 gives accuracies for three baseline models, evaluating over the entire data set. The first baseline simply selects most likely (the *maximum likelihood estimate*; MLE) pronunciation for each instance. The second and third consist of the existing set of disambiguation rules; the server system makes use of the full set of rules, whereas the embedded system excludes POS-tag-based rules, since, as mentioned above, the tagger they depend on cannot currently be run on-device. We see that the existing rules outperform the MLE baseline, but that there is substantial residual error.

5.2. Model accuracies

Table 3 reports accuracies using disjoint training and evaluation sets for all six systems. The two ML systems outperform the rules-only systems (for which the training set is irrelevant), and hybrid ("rules + ML") systems both substantially outperform ML-only systems. As expected, server-based systems outperform comparable embedded systems which lack access to POS tag features. This is particularly pronounced for the ML-only systems. On the other hand, hybridization minimizes this distinction; both hybrid systems perform near ceiling. The best systems (the hybrid systems) obtain a 12.0% absolute and a 92.3% relative error reduction over the worst (embedded rules-only). All six systems are ranked the same by micro- and macro-accuracy, suggesting there is no need to make a distinction between the two metrics on this relatively well-balanced data set.

	Micro	Macro
MLE baseline	.850	.849
Embedded: rules	.869	.863
Server: rules	.893	.890

Table 2: Overall micro- and macro-accuracies for the MLE baseline and two rule-based systems.

	Micro	Macro
Embedded: rules	.870	.867
Server: rules	.890	.886
Embedded: ML	.926	.924
Server: ML	.954	.951
Embedded: rules + ML	.990	.990
Server: rules + ML	.990	.990

Table 3: Evaluation set micro- and macro-accuracies.

5.3. Error analysis

For all six systems, morphosyntactic homographs like *read*, *live*, and *lives* are more challenging than lexical homographs like *bass*. However, the server model performs significantly better on morphosyntactic homographs than the embedded model, presumably due to the presence of POS tag features. One of the most challenging homographs is the mixed homograph *present*. All systems incorrectly predict the verb form [pɹɪˈzɛnt] in place of the noun/adjective [ˈpɹɛzənt] in sentences like "Smith has played Trophy matches for the county from 1993 to *present*." Here the server model incorrectly tags *present* as `VB` (a bare verb), which may contribute to a downstream classification error.

5.4. Limitations

One limitation of all six systems is that they depend on existing tokenization and token classifications, and errors during these steps may propagate. The same is true for POS tagging features. While POS tagging accuracies are quite high in general, morphosyntactic homographs like *present* are instances of a systematic noun-verb ambiguity in English known to be a major source of tagging errors (Toutanova and Manning, 2000). Rules allow such errors to propagate, but discriminative training may also help to correct such errors, i.e., by weighing features so that the evidence for

[3] We do not provide a wider context since some prior work claims that annotators very rarely require additional context to select the appropriate sense (Hearst, 1991; Gale et al., 1992), and single sentences account for the majority of our TTS traffic. In support of this, $< 1\%$ of examples were labeled "ambiguous".

[4] `http://github.com/google/WikipediaHomographData`

[5] These models are stored using a binary wire format based on protocol buffers. At the time of writing our in-production server model for US English has roughly 65k non-zero weights and its serialization is 1.5 MiB in size; our in-production embedded model has 39k non-zero weights and is roughly 800 KiB.

word context overwhelms competing evidence from unreliable POS tag features. It is an open question whether higher-level features, such as those derived from a dependency parse, might help to prevent such errors. Secondly, L_1 regularization is an appropriate remedy for overfitting but it may be suboptimal for controlling size of the embedded model, since the smallest-magnitude weights are not necessarily the least important; feature hashing may be more appropriate for this purpose. Finally, one may suspect that there is substantial redundancy between the existing rules and the ML classifier features in the hybrid systems, but as of yet we do not have a principled method to detect rules subsumed by the ML classifier (or vis versa).

6. Future work

We anticipate that additional languages will pose new challenges and require us to enrich our feature set. For example, consider some challenges posed by Russian and Thai, respectively. In Russian, most homographs are the result of morphologically conditioned stress shifts not indicated in the orthography, and we anticipate that features based on detailed morphological analyses will be required for such homographs. Furthermore, Russian is richly inflected so it may be desirable to use lemmatization or stemming to create equivalence classes for word context features. Thai, on the other hand, is written in a script which lacks capitalization distinctions—making the capitalization feature inapplicable—and does not mark word boundaries (Tesprasit et al., 2003), and thus it may require more sophisticated tokenization schemes. Future work will consider the possibility of replacing or augmenting manually labeled examples with sources of weakly labeled data derived using label propagation (Hearst, 1991), word-aligned bilingual text from a machine translation system—as in Gale et al. (1992)'s approach to word sense disambiguation—or phoneme-aligned transcriptions from a speech recognition system. Furthermore, some recent work on word sense disambiguation employs recurrent neural networks to encode the context (Yuan et al., 2016), allowing for a much wider context window than just the surrounding four tokens used here; we anticipate this technique will also be useful for homograph disambiguation and may even eliminate the need for morphosyntactic features such as POS tags.

7. Conclusion

We have shown that a simple application of machine learning produces significant improvements—in one case, a 12% absolute error reduction—in homograph disambiguation, a key part of high-quality text-to-speech synthesis. Since launching the hybrid server and embedded systems for US English text-to-speech traffic at Google, we have seen a substantial decline in the number of incoming bugs pertaining to homograph disambiguation. Furthermore, we find that once-challenging bug fixes can be completed simply by adding a small number of labeled examples and regenerating the models. Finally, we make our data freely available in the hopes it will encourage future academic research on this understudied problem.

8. Acknowledgements

Thanks to Keith Hall for assistance with the maximum entropy library and early pilot work, and to the many annotators and engineers who participated in data collection and software development.

Braga, D., Coelho, L., and Resende, Fernando Gil V., J. (2007). Homograph ambiguity resolution in front-end design for Portuguese TTS systems. In *INTERSPEECH*, pages 1761–1764.

Ebden, P. and Sproat, R. (2014). The Kestrel TTS text normalization system. *Natural Language Engineering*, 21(3):1–21.

Gale, W. A., Church, K. W., and Yarowsky, D. (1992). Using bilingual materials to develop word sense disambiguation methods. In *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 101–112.

Hall, K. and Sproat, R. (2013). Russian stress prediction using maximum entropy ranking. In *EMNLP*, pages 897–883.

Hearst, M. A. (1991). Noun homograph disambiguation using local context in large text corpora. In *Using Corpora*, pages 185–188.

Silva, D. C., Braga, D., and Resende, Fernando Gil V., J. (2012). A rule-based method for homograph disambiguation in Brazilian Portugese text-to-speech. *Journal of Communications and Information Systems*, 27(1):1–9.

Sproat, R. and Hall, K. (2014). Applications of maximum entropy rankers to problems in spoken language processing. In *INTERSPEECH*, pages 761–764.

Sproat, R., Hirschberg, J., and Yarowsky, D. (1992). A corpus-based synthesizer. In *ICSLP*, pages 563–566.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge University Press, Cambridge.

Tesprasit, V., Charoenpornsawat, P., and Sornlertlamvanich, V. (2003). A context-sensitive homograph disambiguation in Thai text-to-speech synthesis. In *NAACL*, pages 103–105.

Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*, pages 63–70.

Yarowsky, D. (1997). Homograph disambiguation in text-to-speech synthesis. In Jan P. H. van Santen, et al., editors, *Progress in speech synthesis*, pages 157–172. Springer, New York.

Yuan, D., Richardson, J., Doherty, R., Evans, C., and Altendorf, E. (2016). Semi-supervised word sense disambiguation with neural models. In *COLING*, pages 1374–1385.