Hippocratic Abbreviation Expansion

Brian Roark and Richard Sproat
Google, Inc, 79 Ninth Avenue, New York, NY 10011
{roark,rws}@google.com

Abstract

Incorrect normalization of text can be particularly damaging for applications like text-to-speech synthesis (TTS) or typing auto-correction, where the resulting normalization is directly presented to the user, versus feeding downstream applications. In this paper, we focus on abbreviation expansion for TTS, which requires a "do no harm", high precision approach yielding few expansion errors at the cost of leaving relatively many abbreviations unexpanded. In the context of a large-scale, real-world TTS scenario, we present methods for training classifiers to establish whether a particular expansion is apt. We achieve a large increase in correct abbreviation expansion when combined with the baseline text normalization component of the TTS system, together with a substantial reduction in incorrect expansions.

1 Introduction

Text normalization (Sproat et al., 2001) is an important initial phase for many natural language and speech applications. The basic task of text normalization is to convert *non-standard words* (NSWs) — numbers, abbreviations, dates, etc. — into standard words, though depending on the task and the domain a greater or lesser number of these NSWs may need to be normalized. Perhaps the most demanding such application is text-to-speech synthesis (TTS) since, while for parsing, machine translation and information retrieval it may be acceptable to leave such things as numbers and abbreviations unexpanded, for TTS all tokens need to be *read*, and for that it is necessary to know how to pronounce them. Which normalizations are required depends very much on the application.

What is also very application-dependent is the cost of errors in normalization. For some applications, where the normalized string is an intermediate stage in a larger application such as translation or information retrieval, overgeneration of normalized alternatives is often a beneficial strategy, to the extent that it may improve the accuracy of what is eventually being presented to the user. In other applications, such as TTS or typing auto-correction, the resulting normalized string itself is directly presented to the user; hence errors in normalization can have a very high cost relative to leaving tokens unnormalized.

In this paper we concentrate on abbreviations, which we define as alphabetic NSWs that it would be normal to pronounce as their expansion. This class of NSWs is particularly common in personal ads, product reviews, and so forth. For example:

```
home health care svcs stat home health llc
 osceola aquatic ctr stars rating write
  audi vw repair ser quality and customer
```

Each of the examples above contains an abbreviation that, unlike, e.g., conventionalized state abbreviations such as *ca* for *California*, is either only slightly standard (*ctr* for *center*) or not standard at all (*ser* for *service*).

An important principle in text normalization for TTS is *do no harm*. If a system is unable to reliably predict the correct reading for a string, it is better to leave the string alone and have it default to, say, a character-by-character reading, than to expand it to something wrong. This is particularly true in *accessibility* applications for users who rely on TTS for most or all of their information needs. Ideally a navigation system should read *turn on 30N* correctly as *turn on thirty north*; but if it cannot resolve the ambiguity in *30N*, it is far better to read it as *thirty N* than as *thirty Newtons*, since listeners can more easily recover from the first kind of error than the second.

We present methods for learning abbreviation expansion models that favor high precision (incorrect expansions $< 2\%$). Unannotated data is used to collect evidence for contextual disambiguation and to train an abbreviation model. Then a small amount of annotated data is used to build models to determine whether to accept a candidate expan-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 364–369,
Baltimore, Maryland, USA, June 23-25 2014.

sion of an abbreviation based on these features. The data we report on are taken from Google Maps™ and web pages associated with its map entries, but the methods can be applied to any data source that is relatively abbreviation rich.

We note in passing that similar issues arise in automatic spelling correction work (Wilcox-O'Hearn et al., 2008), where it is better to leave a word alone than to "correct" it wrongly.

2 Related work

There has been a lot of interest in recent years on "normalization" of social media such as Twitter, but that work defines normalization much more broadly than we do here (Xia et al., 2006; Choudhury et al., 2007; Kobus et al., 2008; Beaufort et al., 2010; Kaufmann, 2010; Liu et al., 2011; Pennell and Liu, 2011; Aw and Lee, 2012; Liu et al., 2012a; Liu et al., 2012b; Hassan and Menezes, 2013; Yang and Eisenstein, 2013). There is a good reason for us to focus more narrowly. For Twitter, much of the normalization task involves non-standard language such as *ur website suxx brah* (from Yang and Eisenstein (2013)). Expanding the latter to *your website sucks, brother* certainly normalizes it to standard English, but one could argue that in so doing one is losing information that the writer is trying to convey using an informal style. On the other hand, someone who writes *svc ctr* for *service center* in a product review is probably merely trying to save time and so expanding the abbreviations in that case is neutral with respect to preserving the intent of the original text.

One other difference between the work we report from much of the recent work cited above is that that work focuses on getting high F scores, whereas we are most concerned with getting high precision. While this may seem like a trivial trade off between precision and recall, our goal motivates developing measures that minimize the "risk" of expanding a term, something that is important in an application such as TTS, where one cannot correct a misexpansion after it is spoken.

3 Methods

Since our target application is text-to-speech, we define the task in terms of an existing TTS lexicon. If a word is already in the lexicon, it is left unprocessed, since there is an existing pronunciation for it; if a word is out-of-vocabulary (OOV), we consider expanding it to a word in the lexicon. We consider a possible expansion for an abbreviation to be any word in the lexicon from which the abbreviation can be derived by only deletion of letters.[1] For present purposes we use the Google English text-to-speech lexicon, consisting of over 430 thousand words. Given an OOV item (possible abbreviation) in context, we make use of features of the context and of the OOV item itself to enumerate and score candidate expansions.

Our data consists of 15.1 billion words of text data from Google Maps™, lower-cased and tokenized to remove punctuation symbols. We used this data in several ways. First, we used it to bootstrap a model for assigning a probability of an abbreviation/expansion pair. Second, we used it to extract contextual n-gram features for predicting possible expansions. Finally, we sampled just over 14 thousand OOV items in context and had them manually labeled with a number of categories, including 'abbreviation'. OOVs labeled as abbreviations were also labeled with the correct expansion. We present each of these uses in turn.

3.1 Abbreviation modeling

We collect potential abbreviation/full-word pairs by looking for terms that could be abbreviations of full words that occur in the same context. Thus:

the	**svc/service**	center
heating	**clng/cooling**	system
dry	**clng/cleaning**	system

contributes evidence that *svc* is an abbreviation of *service*. Similarly instances of *clng* in contexts that can contain *cooling* or *cleaning* are evidence that *clng* could be an abbreviation of either of these words. (The same contextual information of course is used later on to disambiguate which of the expansions is appropriate for the context.) To compute the initial guess as to what can be a possible abbreviation, a Thrax grammar (Roark et al., 2012) is used that, among other things, specifies that: the abbreviation must start with the same letter as the full word; if a vowel is deleted, all adjacent vowels should also be deleted; consonants may be deleted in a cluster, but not the last one; and a (string) suffix may be deleted.[2] We count a pair of words as 'co-occurring' if they are observed in the same context. For a given context C, e.g., *the__center*, let W_C be the set of words found in that context. Then, for any pair of words u, v, we can assign a pair count based on the count of contexts where both occur:

$$c(u, v) = |\{C : u \in W_C \text{ and } v \in W_C\}|$$

[1] We do not deal here with phonetic spellings in abbreviations such as *4get*, or cases where letters have been transposed due to typographical errors (*scv*).

[2] This Thrax grammar can be found at `http://openfst.cs.nyu.edu/twiki/bin/view/Contrib/ThraxContrib`

blvd	boulevard	rd	road	yrs	years
ca	california	fl	florida	ctr	center
mins	minutes	def	definitely	ste	suite

Table 1: Examples of automatically mined abbreviation/expansion pairs.

Let $c(u)$ be defined as $\sum_v c(u, v)$. From these counts, we can define a 2×2 table and calculate statistics such as the log likelihood statistic (Dunning, 1993), which we use to rank possible abbreviation/expansion pairs. Scores derived from these *type* (rather than *token*) counts highly rank pairs of in-vocabulary words and OOV possible abbreviations that are substitutable in many contexts.

We further filter the potential abbreviations by removing ones that have a lot of potential expansions, where we set the cutoff at 10. This removes mostly short abbreviations that are highly ambiguous. The resulting ranked list of abbreviation expansion pairs is then thresholded before building the abbreviation model (see below) to provide a smaller but more confident training set. For this paper, we used 5-gram contexts (two words on either side) to extract abbreviations and their expansions. See Table 1 for some examples.

Our abbreviation model is a *pair character language model* (LM), also known as a joint multigram model (Bisani and Ney, 2008), whereby aligned symbols are treated as a single token and a smoothed n-gram model is estimated. This defines a joint distribution over input and output sequences, and can be efficiently encoded as a weighted finite-state transducer. The extracted abbreviation/expansion pairs are character-aligned and a 7-gram pair character LM is built over the alignments using the OpenGrm n-gram library (Roark et al., 2012). For example:

```
c:c ϵ:e ϵ:n t:t ϵ:e r:r
```

Note that, as we've defined it, the alignments from abbreviation to expansion allow only identity and insertion, no deletions or substitutions. The cost from this LM, normalized by the length of the expansion, serves as a score for the quality of a putative expansion for an abbreviation.

For a small set of frequent, conventionalized abbreviations (e.g., *ca* for *California* — 63 pairs in total — mainly state abbreviations and similar items), we assign an fixed pair LM score, since these examples are in effect *irregular* cases, where the regularities of the productive abbreviation process do not capture their true cost.

3.2 Contextual features

To predict the expansion given the context, we extract n-gram observations for full words in the TTS lexicon. We do this in two ways. First, we simply train a smoothed n-gram LM from the data. Because of the size of the data set, this is heavily pruned using relative entropy pruning (Stolcke, 1998). Second, we use log likelihood and log odds ratios (this time using standardly defined n-gram counts) to extract reliable bigram and trigram contexts for words. Space precludes a detailed treatment of these two statistics, but, briefly, both can be derived from contingency table values calculated from the frequencies of (1) the word in the particular context; (2) the word in any context; (3) the context with any word; and (4) all words in the corpus. See Agresti (2002), Dunning (1993) and Monroe et al. (2008) for useful overviews of how to calculate these and other statistics to derive reliable associations. In our case, we use them to derive associations between contexts and words occuring in those contexts. The contexts include trigrams with the target word in any of the three positions, and bigrams with the target word in either position. We filter the set of n-grams based on both their log likelihood and log odds ratios, and provide those scores as features.

3.3 Manual annotations

We randomly selected 14,434 OOVs in their full context, and had them manually annotated as falling within one of 8 categories, along with the expansion if the category was 'abbreviation'. Note that these are relatively lightweight annotations that do not require extensive linguistics expertise. The abbreviation class is defined as cases where pronouncing as the expansion would be normal. Other categories included letter sequence (expansion would not be normal, e.g., *TV*); partial letter sequence (e.g., *PurePictureTV*); misspelling; leave as is (part of a URL or pronounced as a word, e.g., *NATO*); foreign; don't know; and junk. Abbreviations accounted for nearly 23% of the cases, and about 3/5 of these abbreviations were instances from the set of 63 conventional abbreviation/expansion pairs mentioned in Section 3.1.

3.4 Abbreviation expansion systems

We have three base systems that we compare here. The first is the hand-built TTS normalization system. This system includes some manually built patterns and an address parser to find common abbreviations that occur in a recognizable context. For example, the grammar covers several hundred city-state combinations, such as *Fairbanks AK*, yielding good performance on such cases.

The other two systems were built using data extracted as described above. Both systems make use of the pair LM outlined in Section 3.1, but differ in how they model context. The first sys-

tem, which we call "N-gram", uses a pruned Katz (1987) smoothed trigram model. The second system, which we call "SVM", uses a Support Vector Machine (Cortes and Vapnik, 1995) to classify candidate expansions as being correct or not. For both systems, for any given input OOV, the possible expansion with the highest score is output, along with the decision of whether to expand.

For the "N-gram" system, n-gram negative log probabilities are extracted as follows. Let w_i be the position of the target expansion. We extract the part of the n-gram probability of the string that is not constant across all competing expansions, and normalize by the number of words in that window. Thus the score of the word is:

$$\mathrm{S}(w_i) \quad = \quad -\frac{1}{k+1} \sum_{j=i}^{i+k} \log \mathrm{P}(w_j \mid w_{j-1} w_{j-2})$$

In our experiments, $k = 2$ since we have a trigram model, though in cases where the target word is the last word in the string, $k = 1$, because there only the end-of-string symbol must be predicted in addition to the expansion. We then take the Bayesian fusion of this model with the pair LM, by adding them in the log space, to get prediction from both the context and abbreviation model.

For the "SVM" model, we extract features from the log likelihood and log odds scores associated with contextual n-grams, as well as from the pair LM probability and characteristics of the abbreviation itself. We train a linear model on a subset of the annotated data (see section 4). Multiple contextual n-grams may be observed, and we take the maximum log likelihood and log odds scores for each candidate expansion in the observed context. We then quantize these scores down into 16 bins, using the histogram in the training data to define bin thresholds so as to partition the training instances evenly. We also create 16 bins for the pair LM score. A binary feature is defined for each bin that is set to 1 if the current candidate's score is less than the threshold of that bin, otherwise 0. Thus multiple bin features can be active for a given candidate expansion of the abbreviation.

We also have features that fire for each type of contextual feature (e.g., trigram with expansion as middle word, etc.), including 'no context', where none of the trigrams or bigrams from the current example that include the candidate expansion are present in our list. Further, we have features for the length of the abbreviation (shorter abbreviations have more ambiguity, hence are more risky to expand); membership in the list of frequent, conventionalized abbreviations mentioned earlier; and some combinations of these, along with bias features. We train the model using standard options with Google internal SVM training tools.

Note that the number of n-grams in the two models differs. The N-gram system has around 200M n-grams after pruning; while the SVM model uses around a quarter of that. We also tried a more heavily pruned n-gram model, and the results are only very slightly worse, certainly acceptable for a low-resource scenario.

4 Experimental Results

We split the 3,209 labeled abbreviations into a training set of 2,209 examples and a held aside development set of 1,000 examples. We first evaluate on the development set, then perform a final 10-fold cross validation over the entire set of labeled examples. We evaluate in terms of the percentage of abbreviations that were correctly expanded (true positives, TP) and that were incorrectly expanded (false positives, FP).

Results are shown in Table 2. The first two rows show the baseline TTS system and SVM model. On the development set, both systems have a false positive rate near 3%, i.e., three abbreviations are expanded incorrectly for every 100 examples; and over 50% true positive rate, i.e., more than half of the abbreviations are expanded correctly. To report true and false positive rates for the N-gram system we would need to select an arbitrary decision threshold operating point, unlike the deterministic TTS baseline and the SVM model with its decision threshold of 0. Rather than tune such a meta-parameter to the development set, we instead present an ROC curve comparison of the N-gram and SVM models, and then propose a method for "intersecting" their output without requiring a tuned decision threshold.

Figure 1 presents an ROC curve for the N-gram and SVM systems, and for the simple Bayesian fusion (sum in log space) of their scores. We can see that the SVM model has very high precision for its highest ranked examples, yielding nearly 20% of the correct expansions without any incorrect expansions. However the N-gram system achieves higher true positive rates when the false

	Percent of abbreviations			
	dev set		full set	
System	TP	FP	TP	FP
TTS baseline	55.0	3.1	40.0	3.0
SVM model	52.6	3.3	53.3	2.6
SVM ∩ N-gram	50.6	1.1	50.3	0.9
SVM ∩ N-gram, then TTS	73.5	1.9	74.5	1.5

Table 2: Results on held-out labeled data, and with final 10-fold cross-validation over the entire labeled set. Percentage of abbreviations expanded correctly (TP) and percentage expanded incorrectly (FP) are reported for each system.

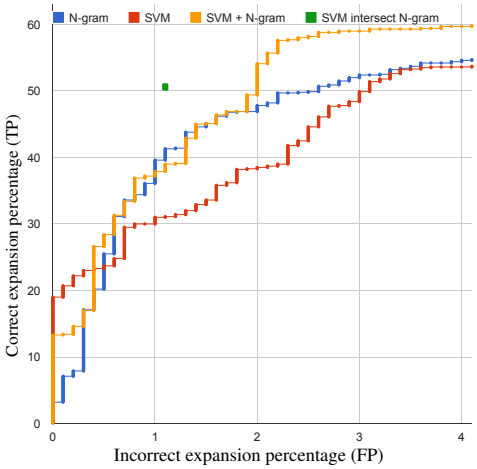

Figure 1: ROC curve plotting true positive (correct expansion) percentages versus false positive (incorrect expansion) percentages for several systems on the development set.

positive rate falls between 1 and 3 percent, though both systems reach roughly the same performance at the SVM's decision threshold corresponding to around 3.3% false positive rate. The simple combination of their scores achieves strong improvements over either model, with an operating point associated with the SVM decision boundary that yields a couple of points improvement in true positives and a full 1% reduction in false positive rate.

One simple way to combine these two system outputs in a way that does not require tuning a decision threshold is to expand the abbreviation if and only if (1) both the SVM model and the N-gram model agree on the best expansion; and (2) the SVM model score is greater than zero. In a slight abuse of the term 'intersection', we call this combination 'SVM intersect N-gram' (or 'SVM $\cap$ N-gram' in Table 2). Using this approach, our true positive rate on the development set declines a bit to just over 50%, but our false positive rate declines over two full percentage points to 1.1%, yielding a very high precision system.

Taking this very high precision system combination of the N-gram and SVM models, we then combine with the baseline TTS system as follows. First we apply our system, and expand the item if it scores above threshold; for those items left unexpanded, we let the TTS system process it in its own way. In this way, we actually reduce the false positive rate on the development set over the baseline TTS system by over 1% absolute to less than 2%, while also increasing the true positive rate to 73.5%, an increase of 18.5% absolute.

Of course, at test time, we will not know whether an OOV is an abbreviation or not, so we also looked at the performance on the rest of the collected data, to see how often it erroneously suggests an expansion from that set. Of the 11,157 examples that were hand-labeled as non-abbreviations, our SVM $\cap$ N-gram system expanded 45 items, which is a false positive rate of 0.4% under the assumption that none of them should be expanded. In fact, manual inspection found that 20% of these were correct expansions of abbreviations that had been mis-labeled.

We also experimented with a number of alternative high precision approaches that space precludes our presenting in detail here, including: pruning the number of expansion candidates based on the pair LM score; only allowing abbreviation expansion when at least one extracted n-gram context is present for that expansion in that context; and CART tree (Breiman et al., 1984) training with real valued scores. Some of these yielded very high precision systems, though at the cost of leaving many more abbreviations unexpanded. We found that, for use in combination with the baseline TTS system, large overall reductions in FP rate were achieved by using an initial system with substantially higher TP and somewhat higher FP rates, since far fewer abbreviations were then passed along unexpanded to the baseline system, with its relatively high 3% FP rate.

To ensure that we did not overtune our systems to the development set through experimentation, we performed 10-fold cross validation over the full set of abbreviations. These results are presented in Table 2. Most notably, the TTS baseline system has a much lower true positive rate; yet we find our systems achieve performance very close to that for the development set, so that our final combination with the TTS baseline was actually slighly better than the numbers on the development set.

5 Conclusions

In this paper we have presented methods for high precision abbreviation expansion for a TTS application. The methods are largely self-organizing, using in-domain unannotated data, and depend on only a small amount of annotated data. Since the SVM features relate to general properties of abbreviations, expansions and contexts, the classifier parameters will likely carry over to new (English) domains. We demonstrate that in combination with a hand-built TTS baseline, the methods afford dramatic improvement in the TP rate (to about 74% from a starting point of about 40%) and a reduction of FP to below our goal of 2%.

Acknowledgments

We would like to thank Daan van Esch and the Google Speech Data Operations team for their work on preparing the annotated data. We also thank the reviewers for their comments.

References

Alan Agresti. 2002. *Categorical data analysis*. John Wiley & Sons, 2nd edition.

Ai Ti Aw and Lian Hau Lee. 2012. Personalized normalization for a multilingual chat system. In *Proceedings of the ACL 2012 System Demonstrations*, pages 31–36, Jeju Island, Korea, July. Association for Computational Linguistics.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779, Uppsala, Sweden, July. Association for Computational Linguistics.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove CA.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Sudesha Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *Int. J. Doc. Anal. Recognit.*, 10:157–174.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Hany Hassan and Arul Menezes. 2013. Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1577–1586.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Max Kaufmann. 2010. Syntactic normalization of Twitter messages. In *International Conference on NLP*.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing SMS: are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 441–448, Manchester, UK, August. Coling 2008 Organizing Committee.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution? Normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 71–76, Portland, Oregon, USA, June. Association for Computational Linguistics.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012a. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1035–1044, Jeju Island, Korea, July. Association for Computational Linguistics.

Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. 2012b. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 526–535, Jeju Island, Korea, July. Association for Computational Linguistics.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of SMS abbreviations. In *IJCNLP*. Papers/pennell-liu3.pdf.

Brian Roark, Michael Riley, Cyril Allauzen, Terry Tai, and Richard Sproat. 2012. The OpenGrm open-source finite-state grammar software libraries. In *ACL*, Jeju Island, Korea.

Richard Sproat, Alan Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287–333.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In *CICLing 2008*, volume 4919 of *LNCS*, pages 605–616, Berlin. Springer.

Yunqing Xia, Kam-Fai Wong, and Wenjie Li. 2006. A phonetic-based approach to Chinese chat text normalization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 993–1000, Sydney, Australia, July. Association for Computational Linguistics.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72.