Reducing infrequent-token perplexity via variational corpora

Yusheng Xie[1,#] **Pranjal Daga**[1] **Yu Cheng**[2] **Kunpeng Zhang**[3] **Ankit Agrawal**[1] **Alok Choudhary**[1]

[1] Northwestern University
Evanston, IL USA

[2] IBM Research
Yorktown Heights, NY USA

[3] University of Maryland
College Park, MD USA

[#] yxi389@eecs.northwestern.edu

Abstract

Recurrent neural network (RNN) is recognized as a powerful language model (LM). We investigate deeper into its performance portfolio, which performs well on frequent grammatical patterns but much less so on less frequent terms. Such portfolio is expected and desirable in applications like autocomplete, but is less useful in social content analysis where many creative, unexpected usages occur (e.g., URL insertion). We adapt a generic RNN model and show that, with variational training corpora and epoch unfolding, the model improves its performance for the task of URL insertion suggestions.

1 Introduction

Just 135 most frequent words account for 50% text of the entire Brown corpus (Francis and Kucera, 1979). But over 44% (22,010 out of 49,815) of Brown's vocabulary are *hapax legomena*[1]. The intricate relationship between vocabulary words and their utterance frequency results in some important advancements in natural language processing (NLP). For example, tf-idf results from rules applied to word frequencies in global and local context (Manning and Schütze, 1999). A common preprocessing step for tf-idf is filtering rare words, which is usually justified for two reasons. First, low frequency cutoff promises computational speedup due to Zipf's law (1935). Second, many believe that most NLP and machine learning algorithms demand repetitive patterns and reoccurrences, which are by definition missing in low frequency words.

1.1 Should infrequent words be filtered?

Infrequent words have high probability of becoming frequent as we consider them in a larger context (e.g., *Ishmael*, the protagonist name in *Moby-Dick*, appears merely once in the novel's dialogues but is a highly referenced word in the discussions/critiques around the novel). In many modern NLP applications, context grows constantly: fresh news articles come out on CNN and New York Times everyday; conversations on Twitter are updated in real time. In processing online social media text, it would seem premature to filter words simply due to infrequency, the kind of infrequency that can be eliminated by taking a larger corpus available from the same source.

To further undermine the conventional justification, computational speedup is attenuated in RNN-based LMs (compared to n-gram LMs), thanks to modern GPU architecture. We train a large RNN-LSTM (long short-term memory unit) (Hochreiter and Schmidhuber, 1997) model as our LM on two versions of *Jane Austen's complete works*. Dealing with 33% less vocabulary in the filtered version, the model only gains marginally on running time or memory usage. In Table 1.1, "Filtered corpus" filters out all the hapax legomena in "Full corpus".

	Full corpus	**Filtered corpus**
corpus length	756,273	751,325
vocab. size	15,125	10,177
running time	1,446 sec	1,224 sec
GPU memory	959 MB	804 MB

Table 1: Filtered corpus gains little in running time or memory usage when using a RNN LM.

Since RNN LMs suffer only small penalty in keeping the full corpus, can we take advantage of this situation to improve the LM?

1.2 Improving performance portfolio of LM

One improvement is LM's performance portfolio. A LM's performance is usually quantified as

[1] Words appear only once in corpus.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 609–615,
Beijing, China, July 26-31, 2015.

perplexity, which is exponentialized negative log-likelihood in predictions.

For our notation, let V_X denote the vocabulary of words that appear in a text corpus $X = \{x_1, x_2, \ldots\}$. Given a sequence $x_1, x_2, \ldots, x_{m-1}$, where each $x \in V_X$, the LM predicts the next in sequence, $x_m \in V_X$, as a probability distribution over the entire vocabulary V (its prediction denoted as p). If $v_m \in V_X$ is the true token at position m, the model's perplexity at index m is quantified as $\exp(-\ln(p[v_m]))$. The training goal is to minimize average perplexity across X.

However, a deeper look into perplexity beyond corpus-wide average reveals interesting findings. Using the same model setting as for Table 1.1, Figure 1 illustrates the relationship between word-level perplexity and its frequency in corpus. In general, the less frequent a word appears, the more unpredictable it becomes. In Table 1.2, the trained model achieves an average perplexity of 78 on filtered corpus. But also shown in Table 1.2, many common words register with perplexity over 1,000, which means they are practically unpredictable. More details are summarized in Table 1.2. The LM achieves exceptionally low perplexity on words such as *<apostr.>s* (*'s*, the possessive case), *<comma>* (, the comma). And these tokens' high frequencies in corpus have promised the model's average performance. Meanwhile, the LM has bafflingly high perplexity on commonplace words such as *read* and *considering*.

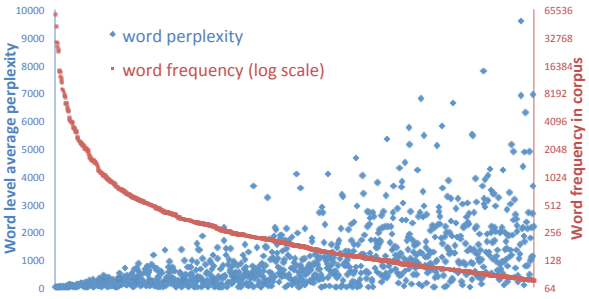

Figure 1: (best viewed in color) We look at word level perplexity with respect to the word frequency in corpus. The less frequent a word appears, the more unpredictable it becomes.

2 Methodology

We describe a novel approach of constructing and utilizing pre-training corpus that eventually reduce LMs's high perplexity on rare tokens. The standard way to utilize a pre-training corpus W is to first train the model on W then fine-tune it on target corpus X. Thanks to availability of text, W can be orders of magnitude larger than X, which makes pre-training on W challenging.

Token	Freq.	Perplexity 1	Perplexity 2
corpus avg.	N/A	78	82
<apostr.>s	4,443	1.1	1.1
of	23,046	4.9	5.0
<comma>	57,552	5.2	5.1
been	3,452	5.4	5.7
read	224	3,658	3,999
quiet	108	6,807	6,090
returning	89	7,764	6,268
considering	80	9,573	8,451

Table 2: A close look at RNN-LSTM's perplexity at word level. "Perplexity 1" is model perplexity based on filtered corpus (c.f., Table 1.1) and "Perplexity 2" is based on full corpus.

A more efficient way to utilize W is to construct variational corpora based on X and W. In the following subsections, we first describe how replacement tokens are selected from a probability mass function (pmf), which is built from W; then explain how the variational corpora variates with replacement tokens through epochs.

2.1 Learn from pre-training corpus

One way to alleviate the impact from infrequent vocabulary is to expose the model to a larger and overarching pre-training corpus (Erhan et al., 2010), if available. Let W be a larger corpus than X and assume that $V_X \subseteq V_W$. For example, if X is Herman Melville's *Moby-Dick*, W can be Melville's complete works. Further, we use $V_{X,1}$ to denote the subset of V_X that are hapax legonema in corpus X; similarly, $V_{X,n}$ (for $n = 2, 3, \ldots$) denotes the subset of V_X that occur n times in X. Many hapax legomena in $V_{X,1}$ are likely to become more frequent tokens in V_W.

Suppose that $x \in V_{X,1}$. Denoted by ReplacePMF(W, V_W, x) in Algorithm 1, we represent x as a probability mass function (pmf) over $\{x'_1, x'_2, \ldots\}$, where each x'_i is selected from $V_W \cap V_{X,n}$ for $n > 1$ using one of the two methods below. For illustration purpose, suppose the hapax legomenon, x, in question is *matrimonial*:

1) e.g., *matrimony*. Words that have very high literal similarity with x. We measure literal similarity using Jaro-Winkler measure, which is an empirical, weighted measure based on string edit

distance. We set the measure threshold very high (> 0.93), which minimizes false positives as well as captures many hapax legonema due to adv./adj., pl./singular (e.g, *-y/-ily* and *-y/-ies*).

2) e.g., *marital* Words that are direct syno/hyponyms to x in the WordNet (Miller, 1995).

getContextAround(x') function in Algorithm 1 simply extracts symmetric context words from both left and right sides of x'. Although the investigated LM only uses left context in predicting word x', context right of x' is still useful information in general. Given a context word c right of x', the LM can learn x''s predictability over c, which is beneficial to the corpus-wide perplexity reduction.

In practice, we select no more than 5 substitution words from each method above. The probability mass on each x'_i is proportional to its frequency in W and then normalized by softmax: $\text{pmf}(x'_i) = \text{freq}(x'_i) / \sum_{k=1}^{5} \text{freq}(x'_k)$. This substitution can help LMs learn better because we replace the un-trainable $V_{X,1}$ tokens with tokens that can be trained from the larger corpus W. In concept, it is like explaining a new word to school kids by defining it using vocabulary words in their existing knowledge.

2.2 Unfold training epochs

Epoch in machine learning terminology usually means a complete pass of the training dataset. many iterative algorithms take dozens of epochs on the same training data as they update the model's weights with smaller and smaller adjustments through the epochs.

We refer to the the training process proposed in Figure 2 (b) as "variational corpora". Compared to the traditional structure in Figure 2 (a), the main advantage of using variational corpora is the ability to freely adjust the corpus at each version. Effectively, we unfold the training into separate epochs. This allows us to gradually incorporate the replacement tokens without severely distorting the target corpus X, which is the learning goal. In addition, variational corpora can further regularize the training of LM in batch mode (Srivastava et al., 2014).

Algorithm 1 constructs variational corpora $X(s)$ at epoch s. Assuming $X(s+1)$ being available, Algorithm 1 appends snippets, which are sampled from W, into $X(s)$ for the sth epoch. For the last epoch $s = S$, $X(S) = X$. As the epoch number increases, fewer and shorter snippets are appended, which alleviates training stress. By fixing an n value, the algorithm applies to all words in $V_{X,n}$.

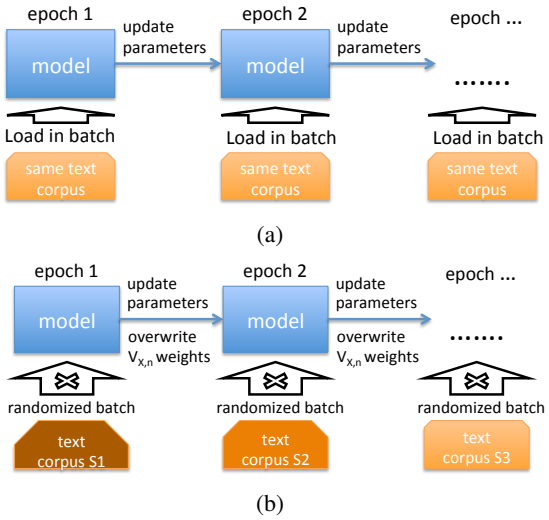

Figure 2: Unfold the training process in units of epochs. (a) Typical flow where model parses the same corpus at each epoch. (b) The proposed training architecture with variational corpora to incorporate the substitution algorithm.

Algorithm 1: Randomly constructs variational corpus at epoch s.

Input: $W, X, V_W, V_X, V_{X,n}, n$, as defined in Section 1.2&2.1,
s, S, current and max epoch number.
Output: $X(s)$, variational corpus at epoch s

1 $X(s) \leftarrow X(s+1)$
2 **for** *each* $x \in V_{X,n}$ **do**
3 $\quad$ **p** $\leftarrow$ ReplacePMF(W, V_W, x)
4 $\quad$ **i** $\leftarrow$ Dirichlet(**p**).generate()
5 $\quad$ **while** $i \leftarrow X$.*getNextIdxOf*(x) **do**
6 $\quad\quad$ $x' \leftarrow$ **i**.draw()
7 $\quad\quad$ $c \leftarrow W$.getContextAround(x')
8 $\quad\quad$ c.substr($\left[0, \text{uniformRnd}\left(0, \frac{S-s}{S}|c|\right)\right]$)
9 $\quad\quad$ $X(s)$.append(c)
10 **return** $X(s)$

In addition, as a regularization trick (Mikolov et al., 2013; Pascanu et al., 2013) , we use a uniform random context window (line 8) when injecting snippets from W into $X(s)$.

Freq.	nofilter	3filter	ptw	vc
10	28,542 (668.1)	23,649 (641.2)	27,986 (1,067.2)	**20,994** (950.9)
100	1,180.3 (21.7)	1,158.2 (19.2)	**735.8** (29.8)	755.8 (31.5)
1K	163.2 (12.9)	163.9 (12.2)	138.5 (14.1)	**137.7** (15.7)
5K	47.5 (3.3)	47.2 (3.1)	**40.2** (3.2)	**40.2** (3.3)
10K	16.4 (0.31)	16.7 (0.29)	14.4 (0.42)	**14.1** (0.41)
40K	7.6 (0.09)	7.6 (0.09)	**7.0** (0.09)	**7.0** (0.10)
all tokens	82.1 (2.0)	77.9 (1.9)	**68.6** (2.1)	68.9 (2.1)
GPU memory	959MB	**783MB**	1.8GB	971MB
running time	1,446 sec	**1,181 sec**	9,061 sec	6,960 sec

Table 3: Experiments compare average perplexity produced by the proposed variational corpora approach and other methods on a same test corpus. Bold fonts indicate best. "Freq." indicates the average corpus-frequency (e.g., Freq.=1K means that words in this group, on average, appear 1,000 times in corpus). Perplexity numbers are averaged over 5 runs with standard deviation reported in parentheses. GPU memory usage and running time are also reported for each method.

Err. type	Context before	True token	LM prediction
False neg.	*<unk>, via, <unk>, banana, muffin, chocolate, ___*	URL to a cooking blog	*recipe*
False neg.	*sewing, ideas, <unk>, inspiring, picture, on, ___*	URL to favim.com	*esty*
False neg.	*nike, sports, fashion, <unk>, women, <unk>, ___*	URL to nelly.com	*macy*
False pos.	*new, york, yankees, endless, summer, tee, <unk>, ___*	*shop*	*<url>*
False pos.	*take, a, rest, from, your, #harrodssale, ___*	*shopping*	*<url>*

Table 4: False positives and false negatives predicted by the model in the Pinterest application. The context words preceding to token in questions are provided for easier analysis[3].

3 Experiments

3.1 Perplexity reduction

We validate our method in Table 3 by showing perplexity reduction on infrequent words. We split Jane Austen's novels (0.7 million words) as target corpus X and test corpus, and her contemporaries' novels[4] as pre-training corpus W (2.7 million words). In Table 3, **nofilter** is the unfiltered corpus; **3filter** replaces all tokens in $V_{X,3}$ by *<unk>*; **ptw** performs naive pre-training on W then on X; **vc** performs training with the proposed variational corpora. Our LM implements the RNN training as described in (Zaremba et al., 2014). Table 3 also illustrates the GPU memory usage and running time of the compared methods and shows that **vc** is more efficient than simply **ptw**.

vc has the best performance on low-frequency words by some margin. **ptw** is the best on frequent words because of its access to a large pre-training corpus. But somewhat to our surprise, **ptw** performs badly on low-frequency words, which we reckon is due to the rare words introduced in W: while pre-training on W helps reduce perplexity of words in $V_{X,1}$ but also introduces additional hapax legomena in $V_{W,1} \setminus V_{X,1}$.

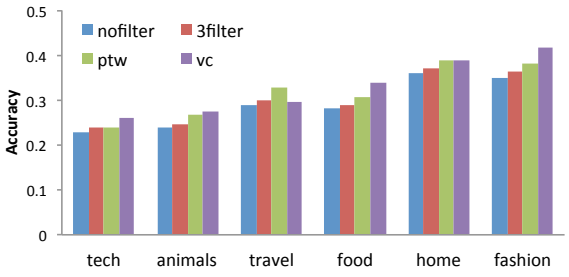

Figure 3: Accuracy of suggested URL positions across different categories of Pinterest captions.

3.2 Locating URLs in Pinterest captions

Beyond evaluations in Table 3. We apply our method to locate URLs in over 400,000 Pinterest captions. Unlike Facebook, Twitter, Pinterest is not a "social hub" but rather an interest-discovery

[3]Favim.com is a website for sharing crafts, creativity ideas. Esty.com is a e-commerce website for trading handmade crafts. Nelly.com is Scandinavia's largest online fashion store. Macy's a US-based department store. Harrod's is a luxury department store in London.

[4]Dickens and the Bronte sisters

site (Linder et al., 2014; Zhong et al., 2014). To maximally preserve user experience, postings on Pinterest embed URLs in a natural, nonintrusive manner and a very small portion of the posts contain URLs.

In Figure 3, we ask the LM to suggest a position for the URL in the context and verify the suggest with test data in each category. For example, the model is presented with a sequence of tokens: *find, more, top, dresses, at, affordable, prices, <punctuation>, visit,* ___ and is asked to predict if the next token is an URL link. In the given example, plausible tokens after *visit* can be either *<http://macys.com>* or *nearest, Macy, <apostr.>s, store*. The proposed **vc** mechanism outperforms others in 5 of the 6 categories. In Figure 3, accuracy is measured as the percentage of correctly suggested positions. Any prediction next to or close to the correct position is counted as incorrect.

In Table 4, we list some of the false negative and false positive errors made by the LM. Many URLs on Pinterest are e-commerce URLs and the vendors often also have physical stores. So in predicting such e-commerce URLs, some mistakes are "excusable" because the LM is confused whether the upcoming token should be an URL (web store) or the brand name (physical store) (e.g, *http://macys.com* vs. *Macy's*).

4 Related work

Recurrent neural network (RNN) is a type of neural sequence model that have high capacity across various sequence tasks such as language modeling (Bengio et al., 2000), machine translation (Liu et al., 2014), speech recognition (Graves et al., 2013). Like other neural network models (e.g., feed-forward), RNNs can be trained using backpropogation algorithm (Sutskever et al., 2011). Recently, the authors in (Zaremba et al., 2014) successfully apply *dropout*, an effective regularization method for feed-forward neural networks, to RNNs and achieve strong empirical improvements.

Reducing perplexity on text corpus is probably the most demonstrated benchmark for modern language models (n-gram based and neural models alike) (Chelba et al., 2013; Church et al., 2007; Goodman and Gao, 2000; Gao and Zhang, 2002). Based on Zipf's law (Zipf, 1935), a filtered corpus greatly reduces the vocabulary size and computation complexity. Recently, a rigorous study (Kobayashi, 2014) looks at how perplexity can be *manipulated* by simply supplying the model with the same corpus reduced to varying degrees. Kobayashi (2014) describes his study from a macro point of view (i.e., the overall corpus level perplexity). In this work, we present, at word level, the correlation between perplexity and word frequency.

Token rarity is a long-standing issue with n-gram language models (Manning and Schütze, 1999). Katz smoothing (Katz, 1987) and Kneser-Ney based smoothing methods (Teh, 2006) are well known techniques for addressing sparsity in n-gram models. However, they are not directly used to resolve unigram sparsity.

Using word morphology information is another way of dealing with rare tokens (Botha and Blunsom, 2014). By decomposing words into morphemes, the authors in (Botha and Blunsom, 2014) are able to learn representations on the morpheme level and therefore scale the language modeling to unseen words as long as they are made of previously seen morphemes. Shown in their work, this technique works with character-based language in addition to English.

5 Acknowledgements

This work is supported in part by the following grants: NSF awards CCF-1029166, IIS-1343639, and CCF-1409601; DOE award DESC0007456; AFOSR award FA9550-12-1-0458; NIST award 70NANB14H012.

6 Conclusions & future work

This paper investigates the performance portfolio of popular neural language models. We propose a variational training scheme that has the advantage of a large pre-training corpus but without using as much computing resources. On low frequency words, our proposed scheme also outperforms naive pre-training.

In the future, we want to incorporate WordNet knowledge to further reduce perplexity on infrequent words.

References

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, Departement D'informatique Et Recherche Operationnelle, and Centre De Recherche. 2000. A neural

probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1899–1907.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Kenneth Church, Ted Hart, and Jianfeng Gao. 2007. Compressing trigram language models with golomb coding. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 199–207.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March.

Nelson Francis and Henry Kucera. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Jianfeng Gao and Min Zhang. 2002. Improving language model size reduction using better pruning criteria. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 176–182, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joshua Goodman and Jianfeng Gao. 2000. Language model size reduction by pruning and clustering. In *Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000, Beijing, China, October 16-20, 2000*, pages 110–113.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6645–6649.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

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

Hayato Kobayashi. 2014. Perplexity on reduced corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 797–806. Association for Computational Linguistics.

Rhema Linder, Clair Snodgrass, and Andruid Kerne. 2014. Everyday ideation: all of my ideas are on pinterest. In *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, pages 2411–2420.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1491–1500, Baltimore, Maryland, June. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1017–1024, New York, NY, USA, June. ACM.

Yee Whye Teh. 2006. A bayesian interpretation of interpolated kneserney. Technical report.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Changtao Zhong, Mostafa Salehi, Sunil Shah, Marius Cobzarenco, Nishanth Sastry, and Meeyoung Cha. 2014. Social bootstrapping: how pinterest and last.fm social communities benefit by borrowing links from facebook. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 305–314.

G.K. Zipf. 1935. *The Psycho-biology of Language: An Introduction to Dynamic Philology*. The MIT paperback series. Houghton Mifflin.