Exploring Word Embeddings for Unsupervised Textual User-Generated Content Normalization

Thales Felipe Costa Bertaglia
ICMC-USP / São Carlos – Brazil
`thales.bertaglia@usp.br`

Maria das Graças Volpe Nunes
ICMC-USP / São Carlos – Brazil
`gracan@icmc.usp.br`

Abstract

Text normalization techniques based on rules, lexicons or supervised training requiring large corpora are not scalable nor domain interchangeable, and this makes them unsuitable for normalizing user-generated content (UGC). Current tools available for Brazilian Portuguese make use of such techniques. In this work we propose a technique based on distributed representation of words (or word embeddings). It generates continuous numeric vectors of high-dimensionality to represent words. The vectors explicitly encode many linguistic regularities and patterns, as well as syntactic and semantic word relationships. Words that share semantic similarity are represented by similar vectors. Based on these features, we present a totally unsupervised, expandable and language and domain independent method for learning normalization lexicons from word embeddings. Our approach obtains high correction rate of orthographic errors and internet slang in product reviews, outperforming the current available tools for Brazilian Portuguese.

1 Introduction

The huge amount of data currently available on the Web allows computer-based knowledge discovery to thrive. The growth in recent years of user-generated content (UGC) – especially the one created by ordinary people – brings forth a new niche of promising practical applications (Krumm et al., 2008). Textual UGC, such as product reviews, blogs, and social network posts, often serves as a base for natural language processing (NLP) tasks. This type of content may be used for business intelligence, targeted marketing, prediction of political election results, analysis of sociolinguistic phenomena, among many other possibilities. Despite its wide range of application, UGC is hard for NLP to handle.

Most of the natural language processing tools and techniques are developed from and for texts of standard language (Duran et al., 2015). From basic components of a NLP-based system, such as taggers, to complex tools aiming to tackle more significant problems, there is a reliance on well structured textual information in order to achieve a proper behavior. However, user generated content does not necessarily follow the structured form of standard language. This type of text is often full of idiosyncrasies, which represent noise for NLP purposes. Beyond the context of NLP, textual UGC may also represent an obstacle for end-users. Specially on social networks, where a domain-specific and context reliant language is used, users not familiar to its particularities may have difficulties to fully grasp the expressed content. Considering the aforementioned problems, it is relevant to identify noises in UGC for purposes which include a simple identification of the noise, its typification, eventually its substitution for another word or expression. These actions aim to both enhance NLP tools performance and facilitate end-user text comprehension, besides to provide an overview of the linguistics practices on the web.

The process of identifying noise and suggesting possible substitutions is known as text normalization. Non-standard words (NSWs) are often regarded as noise, but the precise definition of what constitutes them depends on the application domain. Some examples that might be seen as deviations from standard language and that should be normalized include spelling errors, abbreviations, mixed case words, acronyms, internet slang, hashtags, and emoticons. In general, NSWs are words which properties and meaning cannot be derived directly from a lexicon (Sproat et al., 2001). The term "lexicon" in this context does not necessarily mean the list of words that are formally recognized in a language, but rather

a set of words that are considered treatable by the specific application. Therefore, it is not possible to clearly state what is noise and what should be normalized.

In contrast to standardized language, UGC is often informal, with less adherence to conventions regarding punctuation, spelling, and style (Ling et al., 2013). Therefore, it is considerably noisy, containing *ad-hoc* abbreviations, phonetic substitutions, customized abbreviations, and slang language (Hassan and Menezes, 2013). Considering the specificity of such content, traditional techniques – such as lexicon-based substitution – are not capable to properly handle with many types of noise. Thus, in order to achieve satisfactory results, it is necessary to deeply analyze UGC and develop specific methods aimed at normalizing it.

Conventional string similarity measures (such as edit distance) are not, by themselves, capable of accurately correcting many errors found in UGC. Abbreviations and shorthands found in informal texts, specially on social networks, may contain a large number of edits often resulting in low string similarity – discouraging the use of traditional techniques.

More recently, the interest in techniques based on distributed representations of words (also called word embeddings) has increased. These representations are able to generate continuous numeric vectors of high-dimensionality to represent words. The vectors explicitly encode many linguistic regularities and patterns, as well as syntactic and semantic word relationships (Mikolov et al., 2013a). Words that share semantic similarity are represented by similar vectors. For example, the result of a vector calculation vec("King") - vec("Man") + vec("Woman") is closer to vec("Queen") than to any other word vector. Another strong characteristic of distributed representations is their capability of capturing the notion of contextual similarity – which is essential for textual normalization. The models used for learning these vectors, such as Skip-grams, are totally unsupervised and can be implemented efficiently. In this work, we exploit this set of advantages, combined with lexical similarity measures, in order to capture contextual similarity and learn normalization lexicons based on word embeddings.

2 Related work

Early work handled text normalization as a noisy channel model. This model consists of two components: a source model and a channel model (Shannon, 1948). It assumes that a signal is transferred through a medium and gets corrupted. The source model indicates the canonical form of the signal, and the channel model represents how the signal gets corrupted. (Brill and Moore, 2000) defined the spelling correction problem as finding $argmax_w P(w|s)$, being s the canonical word, which was sent by the source model, and w the received corrupted word. Applying Bayes' Theorem, the noisy channel model is obtained as $argmax_w P(s|w)P(w)$. This model presented significant performance improvements compared to previously proposed models, achieving up to 98% correction accuracy on well-behaved noisy text. However, this approach requires supervised training data for both canonical and corrupted words.

Log-linear models also have been applied as unsupervised statistical models for text normalization. (Yang and Eisenstein, 2013) proposed a model in which the relationship between standard and non-standard words may be characterized by a log-linear model with arbitrary features. The weights of these features can then be trained in maximum-likelihood frameworks. The use of this type of model requires a study of the problem to get the most significant features. From the definition of the features, the training process in conducted to optimize the weights. The advantage of these models is the easy incorporation of new features and the optimization is performed according to an objective function. Although not being highly dependent of resources and context-driven, the log-linear approach requires well-defined features – which are not easily identifiable in UGC. Another disadvantage is the total reliance on statistical observations on the corpus. Hence, the model does not satisfactorily represents the highly semantic specificity of the noise found in UGC, which can occur with low frequency thus not having a significant statistical impact. Considering these issues, this type of model is not enough to deal with generic domain and high context and semantic dependency found is UGC noise.

More recently, social media text normalization was tackled by using contextual graph random walks. (Hassan and Menezes, 2013) proposed a method that uses random walks on a contextual similarity bi-

partite graph constructed from n-gram sequences on large unlabeled text corpus to build a normalization lexicon. They obtained a precision of 92.43% and, using the method as a preprocessing step, improved translation quality of social media text by 6%. (Han et al., 2012) also presented an approach for unsupervised construction of normalization lexicons based on context information. Instead of a graph representation, this approach uses string similarity measures between word within a given context. (Ling et al., 2013) proposed a supervised learning technique for learning normalization rules from machine translations of a parallel corpus of microblog messages. They built two models that learn generalizations of the normalization process – one on the phrase level and the other on the character level. The approach was shown able to improve multiple machine translation systems.

Our technique is most similar to (Sridhard, 2015), since we implement an adaptation of the method presented in the mentioned work. The method proposed by (Sridhard, 2015) aims to learn distributed representations of words to capture the notion of contextual similarity and subsequently learn normalization lexicons from these representations in a completely unsupervised manner. The lexicons are represented as finite-state machines (FSMs) and the process of normalization is performed by transducing the noisy words from the FSMs. Our work makes use of different distributed representation of words, different scoring function for candidate generation and hash structures (dictionaries) instead of FSMs. We also introduce a method for automatically expanding the learned lexicons.

Regarding Brazilian Portuguese, some studies have been performed considering noises in specific domains, such as reviews of products (Duran et al., 2014), and some tools have been developed specifically for that same domain. The normalizer described in (Duran et al., 2015) is, as far as we know, the only tool for text normalization available for Brazilian Portuguese. The proposed lexicon-based normalizer considers that errors found in UGC are divided into six categories: **Common misspellings:** context-free orthographic errors, often phonetically-motivated. **Real-word misspellings:** contextual orthographic errors. Words that are contained in the language lexicon, but are wrong considering the context they appear. **Internet slang**: abbreviations and expressions often used informally by internet users. **Case use (proper names and acronyms)**: proper names and acronyms wrongly or not at all capitalized. **Case use (start of sentence)**: sentences starting with a lower case word. **Glued words**: agglutinated words that should be split. **Punctuation**: wrong use of sentence delimiters.

Since a large part of misspellings found in UGC is phonetically-motivated, (Duran et al., 2015) proposed a phonetic-based speller for correcting such errors. The speller combines edit distance and several specific phonetic rules for Portuguese in order to generate correction candidates. The correction of internet slang and proper name and acronyms capitalization is based on a set of lexicons. Each lexicon contains many pairs of wrong–correct form of words. The correction is performed by looking up the noisy word in the lexicon and substituting it by the correct version. Despite this technique achieving good results in the product review domain, it is not scalable and is too restricted, since there is no form of automatic lexicon-learning. Therefore, it is not suitable for a generic, domain-free normalizer. The results obtained by (Duran et al., 2015) will be further discussed, as they are the main source of comparison for our work.

Another technique specially designed for Brazilian Portuguese is the one proposed by (de Mendonça Almeida et al., 2016). The work presents two approaches for dealing with spelling correction of UGC. The first approach makes use of three phonetic modules, composed by the Soundex algorithm, a grapheme-to-phoneme converter and a set of language-specific phonetic rules. The second one combines grapheme-to-phoneme conversion and a decision tree classifier. The classifier is trained on a corpus of noisy text and employs 14 features (including string and phonetic similarity measures) to identify and correct different classes of orthographic errors. The approach achieves average correction accuracy of 78%, however requires training on an annotated corpus and feature extraction – making it less scalable than an unsupervised technique.

3 Distributed Representation of Words

Distributed representations of words in a vector space, also known as word embeddings, are able to capture lexical, semantic, syntactic, and even contextual similarity between words. This idea has been

recently applied to a wide range of NLP tasks with surprising results (Sridhard, 2015). The Skip-gram model, introduced by (Mikolov et al., 2013a), brought forth an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data. This model, unlike previous ones, does not involve dense matrix multiplications – making the training optimized and efficient.

Figure 1: The Skip-gram model architecture (Mikolov et al., 2013a).
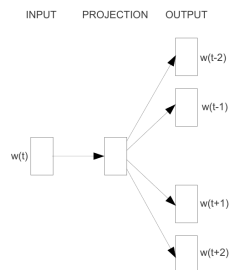
The Skip-gram model, graphically represented in Figure 1, tries to maximize classification of a word based on another word in the same sentence. Each current word is used as input to a log-linear classifier and predicts words in a windows before and after the current one. More formally, the goal of the Skip-gram model is to maximize the average log probability given by:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t) \tag{1}$$

where c is the size of the training context, centered on word w_t (Mikolov et al., 2013b). In practice, word embeddings can be trained using many frameworks that already implement the training process. We used the Gensim [1] framework for Python. The process and parameters we used to train the embeddings will be explained in the next section.

For training distributed representations of words, two sources of data were used in our work. One is a large corpus of product reviews containing both noisy and correct texts. A complete description of the corpus can be found in (Hartmann et al., 2014). The other one is a corpus of Twitter posts (tweets) written in Portuguese. The tweets were extracted using a crawler developed by the authors and the data cannot be distributed due to license restrictions. Most of that content is noisy, considering that the tweets are limited by a maximum of 140 characters and that Twitter is, in general, an informal medium of content generation.

Both corpora had to be preprocessed in order to improve the text quality. First, the Twitter corpus was tokenized using Twokenizer [2], a specific tokenizer for tweets. User names, hashtags and hyperlinks were removed. The product review corpus was tokenized using a simple word tokenizer and predefined tags contained in the reviews (such as "What I liked about this product:") were removed. Next, the sentences were segmented by using the tool provided by (Duran et al., 2015). After segmentation, we obtained 6.8 million sentences from the Twitter corpus and 20 million from the product reviews corpus.

4 Similarity Measures

In order to find candidates for normalizing noisy words, some similarity measures were employed. The cosine distance between two D-dimensional vectors u and v can be used to determine how similar are two word embeddings. It is defined as:

$$\text{cosine similarity} = \frac{\sum\limits_{i=1}^{D} u_i \times v_i}{\sqrt{\sum\limits_{i=1}^{D} (u_i)^2 \times \sum\limits_{i=1}^{D} (v_i)^2}} \tag{2}$$

---

[1] https://radimrehurek.com/gensim/index.html
[2] http://www.cs.cmu.edu/~ark/TweetNLP/#pos

To illustrate the representation power of word embeddings, the following image shows the 25 nearest neighbors (closest cosine similarity) of word 'você' (you) obtained from our word embedding model:

Figure 2: 25 nearest neighbors for word 'você' (you) found in our word embedding model.
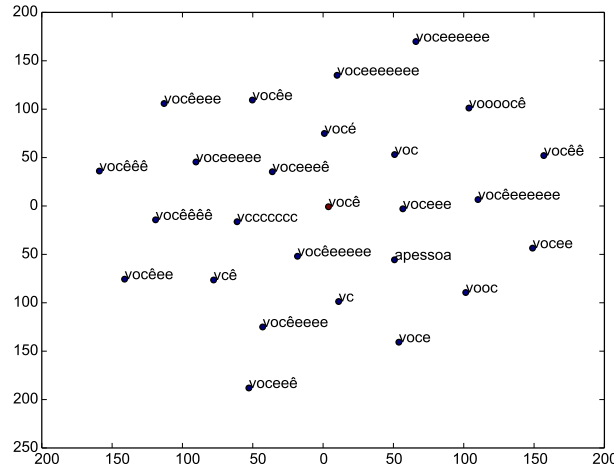
In Figure 2, the canonical form of the word (bold) is located at the center. Around it, there are many noisy variations found in the embedding model. In order to find the canonical form parting from a noisy word, it is also necessary to consider the lexical similarity. We compute its value using an adaptation of the definition presented in (Hassan and Menezes, 2013), as:

$$\text{lexical similarity}(w_1, w_2) = \begin{cases} \frac{LCSR(w_1, w_2)}{MED(w_1, w_2)}, & \text{if} MED(w_1, w_2) > 0 \\ LCSR(w_1, w_2), & \text{otherwise} \end{cases} \quad (3)$$

where w_1 and w_2 are two given words and $MED(w_1, w_2)$ is the modified edit distance between them, calculated as $MED(w_1, w_2) = ED(w_1, w_2) - DS(w_1, w_2)$. $DS(w_1, w_2)$ is the *diacritical symmetry* between w_1 and $w_2 - i.e$, the number of characters from a word that are aligned with versions of itself with diacritical marks in the other word. For instance, the diacritical symmetry between *maçã* (apple) and *maca* (stretcher) is 2. This measure is employed due to the fact that many misspellings in Portuguese happen because of a single diacritic, thus this modification improves the correction of such errors. $LCSR$ refers to Longest Common Subsequence Ratio and is calculated, in our modified version, as:

$$LCSR(w_1, w_2) = \frac{LCS(w_1, w_2) + DS(w_1, w_2)}{\max \text{length}(w_1, w_2)} \quad (4)$$

where LCS refers to Longest Common Subsequence, being usually obtained through a dynamic programming approach. The lexical similarity measure indicates how similar two words are, ranging from 0 (no similarity) to 1 (identical).

5 Learning Normalization Lexicons

Having obtained the set of word embeddings, the method requires a list of canonical words as input. This list may be a lexicon of words recognized by a language, but may also be a list of words considered treatable by the application. We used the UNITEX-PB [3] lexicon for Brazilian Portuguese, removing words with frequency below 100 on Corpus Brasileiro [4] frequency list. This pruning step removes words that probably do not appear in UGC, reducing the lexicon size and improving performance. The next step obtains the K-nearest neighbors, excluding words contained in the lexicon, in the vector space for

---

[3]http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html
[4]http://corpusbrasileiro.pucsp.br/cb/Acesso.html

each canonical word contained in the lexicon. We used K = 25, following (Sridhard, 2015) experiments. This step creates a list of possible corruptions (noisy versions) for every canonical word w_c. Considering the vector space V, the lexicon L and w_c every word $\in L$, the list of noisy versions is obtained as:

$$\text{noisy versions}[w_c] = \forall_{w \in V, V \notin L} \max_k \text{cosine similarity}(w_c, w) \qquad (5)$$

After obtaining the list of noisy versions, it is necessary to invert it, because we must find the canonical version given a noisy word – therefore, the noisy version must index a list of correction candidates to it. In order to do so, we loop through every noisy versions list. We create a new structure to store each noisy word w_n as index to a list of canonical words that contain w_n in its noisy versions list. We then store a tuple $(w_c, score(w_n, w_c))$ as a candidate to correct w_n. The score is calculated as:

$$score(w_n, w_c) = n \times \text{lexical similarity}(w_n, w_c) + (1 - n) \times \text{cosine similarity}(w_n, w_c) \qquad (6)$$

Different from (Sridhard, 2015), our approach considers the cosine similarity at the score function, because some corrections have low lexical similarity but appear in the same context, specially for abbreviations – for instance, *d+* as abbreviation for *demais* (too much). The term n is a value between 0 and 1 that indicates the weight (or importance) given to the lexical similarity. In our experiments, n was set as 0.8. The learned lexicon is stored as a hash structure, indexed by w_n, where each entry contains the list of candidate corrections for the respective noisy word. To determine the best candidate to correct w_n, we simply compute $\max(score(w_n, w_c))$. Algorithm 1 summarizes the lexicon learning process.

Input: list of canonical words L, word embedding model V, number of K nearest neighbors
Output: learned lexicon T where $T[w_n]$ is the list of candidates for correcting noisy word w_n
for *each $w \in L$* **do**
    for *each $v \in V$* **do**
        if $v \notin L$ **then**
            compute cosine similarity(w, v)
        end
        store top K neighbors in $C(w)$
    end
end
for *each $w \in C$* **do**
    for *each $w' \in C[w]$* **do**
        $T[w'] \Leftarrow (w, score(w', w))$
    end
end

Algorithm 1: Unsupervised Lexicon Learning

5.1 Expanding the Learned Lexicon

Given that only the K-nearest neighbors of a canonical word are added to the lexicon, some very infrequent noisy words may not be a top neighbor of any canonical word. In this case, the approach presented above cannot find any correction candidates. In order to solve this issue, we added an expansion step to the method. If a given noisy w_n is not in the learned lexicon, that is, $w_n \notin T$, then we find the canonical word from L which is most similar to w_n, and add it as a correction candidate. Thus, the expansion step can be defined as:

$$\text{If } w_n \notin T : T[w_n] \Leftarrow \max_{w_c \in L} \text{lexical similarity}(w_n, w_c) \qquad (7)$$

Despite its simplicity, the expansion step improved the overall correction recall.

5.2 Representing Context

Thus far, our framework has scored candidates deterministically, based solely on lexical information. In this case, correction candidates will always be the same for a given noisy word, independent from the

context it appears. This idea might be enough for correcting internet slang, since equal abbreviations are seldom used for different words. However, ignoring contextual information makes it impossible to handle contextual orthographic errors (also called real-word errors or RWEs). This type of error occurs when a given word is in the recognized lexicon (so it is not *per se* a noisy word) but is incorrect due to the context it appears. In Portuguese, cases of real-word errors caused by missing a single diacritic are frequent and can only be corrected by considering the context. In order to do so, we used a language model (LM) (with trigrams) trained on a Wikipedia sample consisting of 3841834 sentences. We trained our LM using the KenLM framework, which employs modified Kneser-Ney smoothing for estimation (Heafield, 2013). The estimated trigram probabilities were normalized to fit the range of the other measures and then added to the scoring function. The correction of RWEs relies only on similarity measures and on the language model, because it maps a canonical (but inadequate to the context) word to another canonical word – therefore the learned lexicon based on word embeddings does not include it.

6 Evaluation

In order to evaluate the proposed method and compare it to already existing tools for Brazilian Portuguese, two annotated samples of product reviews written by users were used. Each sample contains 60 reviews, with every error manually annotated by a specialist. The annotation considers the six categories of noise proposed by (Duran et al., 2015), but our technique can only be applied to the correction of orthographic errors and internet slang.

First, we conducted experiments in order to determine the best word embedding model. We trained both Skip-gram and Continuous bag-of-words (Cbow) models, implemented in Gensim, to learn the embeddings. We used a context window of size 5, *i.e* 2 words before and 2 after the center, and used hierarchical sampling for reducing the vocabulary size during training, considering only words with a minimum count of 10 occurrences. We generated embeddings with 100, 300 and 500 dimensions. For each dimension size, we trained the embeddings on two different sets of data: the first one, referred as Noisy, is exactly as described in Section 3, and the second one, referred as Hybrid, includes an additional dataset containing 38 million sentences from Wikipedia. Table 1 shows the recall measure obtained for the correction of orthographic errors (O) (without the LM probabilities) and internet slang (I) (with the best setup) using each embedding model trained.

Table 1: Results for error correction obtained from each word embedding model

| Model | Dimensions | | | | | |
| | 100 | | 300 | | 500 | |
	O	I	O	I	O	I
Cbow Hybrid	67.2%	64.5%	72.3%	64.5%	72.1%	64.5%
Cbow Noisy	70.7%	54.8%	74.7%	64.5%	76.0%	64.5%
Skip-gram Hybrid	63.8%	77.4%	74.2%	71.0%	76.0%	71.0%
Skip-gram Noisy	77.3%	64.5%	78.6%	67.7%	**78.6%**	**77.4%**

The Skip-gram model with 500 dimensions showed the best results. Therefore, further experiments were conducted using these embeddings. In order to better evaluate the framework, we performed experiments using three different learned lexicons models. The only difference between them is the word embedding model employed. The first one, referred as *Noisy*, uses embeddings trained on data preprocessed exactly as described in Section 3. The second model, referred as *Clean*, uses embedding trained on data with an additional preprocessing step: every symbol that is not either a letter or a number is removed. Both are Skip-gram models with 500 dimensions. The third model is an *Ensemble* containing the previous ones. Its output is defined as $max(output(Clean), output(Noisy))$. We have found empirically that the *Noisy* model is better for internet slang correction, the *Clean* model is better for orthographic errors and the *Ensemble* joins the best of both. In Tables 2 and 3 we compare the three lex-

icon models with UGCNormal, the tool proposed by (Duran et al., 2015). The tables follow the format $X/Y = Z$, being X the total of corrections performed by each model, Y the total of annotated errors in the sample and Z the obtained recall measure.

Table 2: Results for orthographic errors correction

Model	Corrections
Noisy	129/164 = 78.6%
Clean	135/164 = 82.3%
Ensemble	137/164 = 83.5%
Ensemble+Expansion Step	149/164 = 90.9%
Ensemble+Expansion Step+LM	**151/164 = 92.1%**
Expansion Step+LM (RWEs)	**84/115 = 73.0%**
UGCNormal	137/164 = 83.5%
UGCNormal (RWEs)	39/115 = 33.9%

Table 3: Results for internet slang correction

Model	Corrections
Noisy	20/31 = 64.5%
Clean	17/31 = 54.8%
Ensemble	22/31 = 71.0%
Ensemble+Expansion Step	23/31 = 74.2%
Ensemble+Expansion Step+LM	**24/31 = 77.4**%
UGCNormal	19/31 = 61.3%

The results show that the ensemble model with expansion step and context representation (LM) outperforms every other model, including UGCNormal. For RWEs, the combination of expansion step and language model outperforms UGCNormal by a large margin.

7 Conclusion

We presented an unsupervised method for learning normalization lexicons based on similarity measures and distributed representation of words. Our approach does not require sets of rules or domain-specific lexicons, but only a lexicon containing a list of canonical words. Therefore, it can be easily expanded and adapted according to the needs of each specific application. We trained distributed representations using the Skip-gram model on a large corpus of both Twitter posts and product reviews. The results indicate that the method for both internet slang and orthographic error correction surpasses the results obtained from already existing tools for Brazilian Portuguese. The expansion step, despite being simple, improved the correction significantly. As future work, different methods of context representation must be explored. Methods for dealing with multiword expressions, such as acronyms for internet slang, must be added to our framework. Despite being rare in Brazilian Portuguese, such language constructs are commonly find on internet slang, mainly derived from English. Correcting more categories of noise, either by expanding this technique or investigating new approaches, is the following step for this project.

References

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gustavo Augusto de Mendonça Almeida, Lucas Avanço, Magali Sanches Duran, Erick Rocha Fonseca, Maria das Graças Volpe Nunes, and Sandra Maria Aluísio. 2016. Evaluating phonetic spellers for user-generated content in brazilian portuguese. In *Computational Processing of the Portuguese Language: 12th International Conference, PROPOR 2016, Tomar, Portugal, July 13-15, 2016, Proceedings*, pages 361–373. Springer International Publishing.

Magali Sanches Duran, Lucas Avanço, Sandra M Aluisio, Thiago A S Pardo, and M Graças Volpe Nunes. 2014. Some issues on the normalization of a corpus of products reviews in Portuguese. *Proceedings of the EACL 2014 9th Web as Corpus Workshop (WAC-9)*, pages 22–27.

Magali Sanches Duran, Lucas Avanço, and M Graças Volpe Nunes. 2015. A Normalizer for UGC in Brazilian Portuguese. *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text*, pages 38–47.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *EMNLP-CoNLL 2012 - 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Proceedings of the Conference*, pages 421–432.

Nathan Hartmann, Lucas Avanço, Pedro Balage, Magali Duran, Maria Das Graças Volpe Nunes, Thiago Pardo, and Sandra Aluísio. 2014. A large corpus of product reviews in portuguese: Tackling out-of-vocabulary words. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

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

Kenneth Heafield. 2013. Scalable modified kneser-ney language model estimation. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

John Krumm, Nigel Davies, and Chandra Narayanaswami. 2008. User-generated content. *Ieee Cs*, 7:10–11.

Wang Ling, Chris Dyer, W. Alan Black, and Isabel Trancoso. 2013. Paraphrasing 4 Microblog Normalization. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. *Nips*, pages 1–9.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. *ArXiv Preprint*.

Claude E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423.

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

K. R. Vivek Sridhard. 2015. Unsupervised Text Normalization Using Distributed Representations of Words and Phrases. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 8–16.

Yi Yang and Jacob Eisenstein. 2013. A Log-Linear Model for Unsupervised Text Normalization. *Emnlp*, (October):61–72.