Garbled-Word Embeddings for Jumbled Text

Gianluca Sperduti[1], Alejandro Moreo[1] and Fabrizio Sebastiani[1]

[1] *Istituto di Scienza e Tecnologie dell'Informazione*
Consiglio Nazionale delle Ricerche
Via Giuseppe Moruzzi 1
Pisa, Italy, 56124

Abstract

"Aoccdrnig to a reasrech at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny itmopnrat tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe". We investigate the extent to which this phenomenon applies to computers as well. Our hypothesis is that computers are able to learn distributed word representations that are resilient to character reshuffling, without incurring a significant loss in performance in tasks that use these representations. If our hypothesis is confirmed, this may form the basis for a new and more efficient way of encoding character-based representations of text in deep learning, and one that may prove especially robust to misspellings, or to corruption of text due to OCR. This paper discusses some fundamental psycho-linguistic aspects that lie at the basis of the phenomenon we investigate, and reports on a preliminary proof of concept of the above idea.

Keywords

Garbled-Word Embeddings, Garbled Words, Misspellings, Distributional Semantic Models

1. Introduction

Since 2003, the sentence quoted in the abstract of the present paper has been circulating around the Internet, and has become fairly popular in social networks and forums. Even though there is no evidence whatsoever of any such research at Cambridge University, a lot of psycho-linguistic literature [1, 2, 3, 4, 5] has shown that, though at the cost of sacrificing reading speed (a cost that depends on the specific words that are affected), humans can actually understand *garbled* (a.k.a. *jumbled*) *text*, which we here define as text containing words affected by character reshuffling, as long as the first and last letters of each word stay in place. In this paper we put forth our conjecture that, if humans can make sense of garbled text, it is likely that computerized distributional semantic models can do so as well. Despite the fact that some previous work has suggested that this may not be the case [6, 7], we call into question the way surface forms are customarily represented in this field, and propose a new mechanism that natively disregards character order.

One of the goals of our investigation is furthering our understanding of the connections

IIR 2021 – 11th Italian Information Retrieval Workshop, September 13–15, 2021, Bari, Italy
gianluca.sperduti@isti.cnr.it (G. Sperduti); alejandro.moreo@isti.cnr.it (A. Moreo); fabrizio.sebastiani@isti.cnr.it (F. Sebastiani)
0000-0002-4287-8968 (G. Sperduti); 0000-0002-0377-1025 (A. Moreo); 0000-0003-4221-6427 (F. Sebastiani)

between the capabilities of humans and those of computers. However, the vectorial representations that derive from this investigation, called *garbled-word embeddings*, have an applicative impact too, since they could also be applied to building resiliency to misspellings into computerized models of text, to automatic misspelling correction, and to handling out-of-vocabulary terms, and could hopefully set the foundations of a new, more efficient modality of representing character-based textual inputs for deep neural models. While we plan to cope with all these aspects in our future research, we devote this short paper to studying the viability of garbled-word embeddings.

This paper is organized as follows. In Section 2 we review and discuss the main works devoted to the study of the above-mentioned psycho-linguistic phenomena, and the (few) computational approaches that have derived. In order to verify the conjecture we have described, we subsequently devise a transformation of the word surface form (that we dub *BE-sorting* – Section 3) whose output is consistent with all possible "garbled variants" of the same word. We then learn embeddings for BE-sorted words, by using any of the publicly available techniques (we here use `word2vec` [8]), from a corpus of texts in which each word occurrence has been replaced by its corresponding BE-sorted version. We then compare the performance of these garbled-word embeddings, across a series of downstream tasks and standard benchmarks for them, with that of standard word embeddings. The experiments we have carried out are presented and discussed in Section 4. We sketch future work in Section 5.

2. Related work

How transposed letters and misspellings affect our reading performance has long been studied in the field of psycho-linguistics [1, 2, 3, 4, 5] and, more recently, also in the area of NLP [6, 7, 9, 10].

The literature distinguishes two different types of word corruption, i.e., (a) corruption due to transpositions only (whereby an anagram of the original word is formed – this is what we have called *garbling*), and (b) corruption that also involves letter insertion and/or deletion and/or replacement. In their seminal work, Rayner et al. [1] show that humans are able to read (and understand) corrupted text with good reading speed (which is proportional to the inverse of cognitive effort), but that this speed depends on which type of word corruption is involved, i.e., humans deal with type (a) ("garbling") much more easily than with type (b). In this paper we only consider word garbling. Concerning garbled text, Rayner et al. [1] find that reading speed decays by 12% when the transposed letters do not involve the first and last letters of the word, by 26% when the transposed letters are the first and the last one, and by 36% when the transposition involves the first and second letters of the word.

Many studies have been devoted to identifying the factors and conditions that impact on the difficulty of reading garbled words. Andrews [3] finds that a garbled word is easier to process when it is a highly frequent one, since, when reading, words that are common in language surprise us the least, and involve thus a smaller cognitive effort to make sense of when they occur in garbled form [4]. Many authors seem to agree that the *neighbourhood size* n of a garbled word (the neighbourhood of a given word being the set containing all valid words that can be obtained by replacing a single letter) is a good proxy for estimating the difficulty of processing it, with higher values of n typically implying higher reading difficulty [3]. English terms tend to display higher values of n on average than many other languages [5], thus making it one of the most challenging languages for reading garbled words. We use English as our target language in this study.

In the field of NLP, several studies have been carried out in order to assess the resiliency of embedding-generation techniques and pre-trained language models to misspellings and garbled words. Heigold et al. [6] test the impact that different types of noise have on the performance of convolutional and recurrent neural architectures equipped with different input modes (character-based and byte-pair encoding) in morphological tagging and translation tasks, concluding that all models are highly sensitive to (even subtle) perturbations in the text, but that such degradation in performance can be partially countered by reproducing the same noise conditions in the training data. Yang and Gao [7] show that even stronger models like BERT [11] cannot elude the catastrophic deterioration in performance due to the presence of corrupted text. Nguyen and Grieve [9] compare the performance of word embeddings generated from corrupted words using `word2vec`'s *skip-gram with negative sampling* (SGNS – [8]) and `fastText` [12, 13], showing that SGNS tends to perform better than `fastText` in the majority of intrinsic tasks[1], despite the fact that SGNS is a spelling-agnostic method.[2] The authors also note that both methods tend to perform fairly well in tasks characterized by the presence of emotion-laden, intentionally elongated words (e.g., "gooood"), and they conjecture that this may be due to the fact that such intentional misspellings tend to occur in specific and controlled contexts. Other work has focused on creating embeddings that are resilient to misspellings. Arguably, the most notable such example is *Misspelling-Oblivious Word Embeddings* (MOEs) [10]. MOEs are based on the `fastText` algorithm extended with a redesigned loss function that implements a distance between correct and misspelled words. MOEs have been tested in both "intrinsic" and "extrinsic" tasks at various levels of corruption, with notable results in highly corrupted text. Differently from this approach, we do not attempt to create word embeddings for the (potentially many) altered surface forms of the words in the vocabulary, but instead BE-sort each word, thus "collapsing" all possible such variations into a single canonical form which is then handled as any standard word by any off-the-shelf word-embedding generation technique.

3. BE-sorting

The method we propose is very easy, and comes down to a simple pre-processing of the text input. Given a word $w = [c_1, c_2, \ldots, c_n]$, in which c_i denotes the character at position i, we *BE-sort* it, i.e., we map it into an artificial token $R(w) = [c_1, \mathrm{sort}([c_2, \ldots, c_{n-1}]), c_n]$ in which the b̲eginning (B) and e̲nd (E) characters are left at their original position while the other ones (which we here call the "middle" ones) are sorted in alphabetical order. (Any deterministic sorting method other than alphabetical would do as well, though.) For example, given the input word "embedding" (short for $[e, m, b, e, d, d, i, n, g]$), the function R will return the BE-sorted

[1]Consistently with this literature, we call *intrinsic* (resp., *extrinsic*) those tasks in which words (resp., documents) are the objects of study, and which are directly (resp., indirectly) affected by how we handle words; an example is word analogy (resp., text classification).

[2]In particular, the authors found out that `fastText` performs better in tasks where some characters have been deleted, while SGNS performs better in all other tasks, and especially so in tasks having to do with semantic analogy.

token “ebddeimng”; any garbled variant of “embedding” in which the B and E characters are in their original position (e.g., “ebdimnedg”, “edmnbdieg”) will result in the same BE-sorted token.

We apply the R function to replace all word occurrences in the training corpus with their BE-sorted versions, and then use, in the standard way, any standard function for generating word embeddings. During test, we BE-sort all word occurrences in the test corpus and use the embeddings generated at training time. Note that the sorting function produces a deterministic and unique version of the text corpus, irrespective of whether the original text was garbled or not. Should our hypothesis be correct, the word embeddings learned from the BE-sorted training corpus would behave no worse than a set of embeddings learned from the original (non-BE-sorted) corpus.

4. Experiments

We choose the British National Corpus (BNC)[3] as our corpus for training word embeddings. From the BNC, we generate different sets of embeddings:

- **Garbled($x\%$)**, obtained by randomly picking $x\%$ of the word occurrences in the corpus, garbling them, and generating embeddings from the resulting (garbled or non-garbled) words; we generate five different instances, i.e., for $x \in \{5, 10, 50, 100\}$); note that the $x = 0$ version corresponds to embeddings generated from the original version of the BNC, while for the $x = 100$ version all the words from which the embeddings are generated are garbled.
- **BE-sorted**, obtained by BE-sorting all word occurrences in the corpus and generating embeddings from the resulting tokens;
- **Full-sorted**, obtained by sorting *all* characters in the word alphabetically, for each word occurrence in the corpus; this is the same as the BE-sorted version, but for the fact that the B and E characters of the word are not necessarily left at their original positions;
- **RandEmbeds**, a set of embeddings (one for each word in the original corpus) randomly generated and not optimized any further; this set establishes a lower-bound baseline.

For each variant of our dataset, we lower-case all text and then generate word embeddings using `word2vec`'s SGNS [8], using the same hyper-parameter values in all cases. [4] In order to compensate for fluctuations in results due to randomness, we carried out 10 runs for each model using different seeds. Table 1 reports the results of our experiments in using the sets of embeddings learned from each corpus across a well-established battery of intrinsic-task benchmarks[5], comprising 17 different tasks dealing with semantic categorization, word similarity, word relatedness, and word analogy (see [14] for further details).

[3]The British National Corpus was originally created by Oxford University, and contains 100-million-words worth of text from various genres from the later part of the 20th century, including spoken, fiction, magazines, newspapers, and academic. The corpus is available at http://www.natcorp.ox.ac.uk/corpus/index.xml .

[4]We use the `Gensim` implementation with hyper-parameter values `min_count`=1, `max_vocab_size`=None (i.e., without limit), `window`=5, `vector_size`=300, `sample`=6e-5, `alpha`=0.03, `min_alpha`=0.0007, `negative`=20, `sg`=1. We leave the other hyper-parameters at their default values.

[5]https://github.com/kudkudak/word-embeddings-benchmarks

Table 1
Performance evaluation of different sets of embeddings on 17 intrinsic-task benchmarks, grouped according to task (2nd row) and evaluation measure (3rd row).

	AP	BLESS	Battig	ESSLLI2c	ESSLLI2b	ESSLLI1a	MEN	WS353	WS353R	WS353S	MTurk	SL999	RW	RG65	Google	MSR	SE2012
	Categorization						Relatedness					Similarity			Analogy		
	Purity						Correlation					Correlation			Accuracy		
Garbled(0%)	.618	**.835**	**.376**	**.662**	**.765**	**.847**	.725	.635	.588	.682	**.553**	**.329**	**.160**	.782	**.262**	**.015**	.148
Garbled(5%)	**.640**	.818	.376	.653	.747	.836	**.728**	.635	.590	.683	.548	.324	.144	**.788**	.240	.012	**.153**
Garbled(10%)	.628	.819	.372	.640	.750	.822	.726	**.640**	.596	.680	.536	.320	.133	**.788**	.220	.009	.147
Garbled(50%)	.593	.804	.344	.600	.710	.795	.713	.627	**.606**	.662	.520	.267	.054	.735	.087	.002	.142
Garbled(100%)	.333	.539	.192	.566	.625	.663	.439	.253	.205	.288	.030	.140	.134	.377	.002	.000	.069
BE-sorted	.622	.833	.374	.644	.745	.841	.719	**.640**	.594	**.685**	.549	.324	.157	.785	.241	.015	.150
Full-sorted	.499	.626	.328	.515	.675	.659	.211	.249	.295	.250	.221	.124	.132	.049	.196	.009	.080
RandEmbeds	.159	.230	.092	.378	.525	.432	-.018	.127	.178	.048	-.074	.010	-.038	.006	.000	.000	.011

Interestingly, the results obtained with our method (**BE-sorted**) are, by and large, very similar to the ones obtained on the original corpus (**Garbled(0%)**), and almost always superior to those obtained by **Garbled(5%)**, thus confirming our initial hypothesis. When character sorting is not used, performance seems to deteriorate as the fraction of garbled word occurrences increases. The results also clearly indicate that **Full-sorted** fares worse than **BE-sorted**, thus bringing empirical support to the intuition according to which, for computer-based distributional semantic models, as well as for humans, the first and the last letters should remain in place in order to achieve comparable performance. We are currently investigating this aspect in greater detail.

5. Conclusions

Somehow surprisingly, humans can effectively read garbled words, provided that their first and last letters stay in place. While this phenomenon has long lured the attention of psycholinguists, authors of recent computational experiments have argued that the performance of computational models, unlike that of humans, deteriorates noticeably in the presence of garbled input. However, we hypothesize that computational models can be made robust to garbled text too, and argue that the key to doing this is devising word representation mechanisms that *natively* disregard character order within words. As a preliminary proof of concept, we have shown that it is indeed possible to learn word embeddings from text in which we simply sort the middle letters of each word (thus intentionally getting rid of any character order information) with no substantial difference in performance. We are currently investigating character-based representations for deep learning that implement this intuition.

References

[1] K. Rayner, S. White, R. Johnson, S. Liversedge, Raeding wrods with jubmled lettres: There is a cost, Psychological Science 17 (2006) 192–3.

[2] L. X. McCusker, P. B. Gough, R. G. Bias, Word recognition inside out and outside in, Journal of Experimental Psychology: Human Perception and Performance 7 (1981) 538.

[3] S. Andrews, Lexical retrieval and selection processes: Effects of transposed-letter confusability, Journal of Memory and Language 35 (1996) 775–800.

[4] A. F. Healy, Detection errors on the word the: Evidence for reading units larger than letters, Journal of Experimental Psychology: Human Perception and Performance 2 (1976) 235.

[5] V. Marian, J. Bartolotti, S. Chabal, A. Shook, Clearpond: Cross-linguistic easy-access resource for phonological and orthographic neighborhood densities, PLOS ONE 7 (2012) e43230.

[6] G. Heigold, S. Varanasi, G. Neumann, J. van Genabith, How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse?, in: Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track), Boston, US, 2018.

[7] R. Yang, Z. Gao, Can machines read jmulbed senetcnes? (2019). URL: https://runzhe-yang.science/demo/jumbled.pdf.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS 2013), Lake Tahoe, US, 2013, pp. 3111–3119.

[9] D. Nguyen, J. Grieve, Do word embeddings capture spelling variation?, in: Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, ES, 2020, pp. 870–881.

[10] A. Piktus, N. B. Edizel, P. Bojanowski, E. Grave, R. Ferreira, F. Silvestri, Misspelling-oblivious word embeddings, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, Minneapolis, US, 2019, pp. 3226–3234.

[11] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019), Minneapolis, US, 2019, pp. 4171–4186.

[12] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146. doi:10.1162/tacl_a_00051.

[13] E. Grave, T. Mikolov, A. Joulin, P. Bojanowski, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017), Valencia, ES, 2017, pp. 427–431.

[14] A. Lenci, M. Sahlgren, P. Jeuniaux, A. C. Gyllensten, M. Miliani, A comprehensive comparative evaluation and analysis of distributional semantic models, arXiv preprint arXiv:2105.09825 (2021).