applied sciences

MDPI

Article

Towards Robust Word Embeddings for Noisy Texts

Yerai Doval [1,*], Jesús Vilares [2] and Carlos Gómez-Rodríguez [2]

1 Grupo COLE, Escola Superior de Enxeñaría Informática, Universidade de Vigo, 36310 Vigo, Spain
2 Universidade da Coruña, CITIC. Grupo LyS, Departamento de Ciencias da Computación e Tecnoloxías da Información, 15071 A Coruña, Spain; jesus.vilares@udc.es (J.V.); carlos.gomez@udc.es (C.G.-R.)
* Correspondence: yerai.doval@uvigo.es

Received: 26 August 2020; Accepted: 28 September 2020; Published: 1 October 2020

Abstract: Research on word embeddings has mainly focused on improving their performance on standard corpora, disregarding the difficulties posed by noisy texts in the form of tweets and other types of non-standard writing from social media. In this work, we propose a simple extension to the skipgram model in which we introduce the concept of bridge-words, which are artificial words added to the model to strengthen the similarity between standard words and their noisy variants. Our new embeddings outperform baseline models on noisy texts on a wide range of evaluation tasks, both intrinsic and extrinsic, while retaining a good performance on standard texts. To the best of our knowledge, this is the first explicit approach at dealing with these types of noisy texts at the word embedding level that goes beyond the support for out-of-vocabulary words.

Keywords: natural language processing; semantics; word embeddings; noisy texts; social media

1. Introduction

Continuous word representations, also known as word embeddings, have been successfully used in a wide range of NLP tasks such as dependency parsing [1], information retrieval [2], POS tagging [3], or Sentiment Analysis (SA) [4]. A popular scenario for NLP tasks these days is social media platforms such as Twitter [5–7], where texts are usually written without following the standard rules, containing varying levels of noise in the form of spelling mistakes ('socisl' for 'social'), phonetic spelling of words ('dat' for 'that'), abbreviations for common phrases ('tbh' for 'to be honest'), emphasis ('yessss' as an emphatic 'yes') or incorrect word segmentations ('noway' for 'no way'). However, the most commonly-used word embedding approaches do not take these phenomena into account [8–10], and we instead rely on their implicit capacity to cope with non-standard words provided a large enough amount of varied training text, such as in [11].

Another possibility to tackle non-standard texts would be to apply some preprocessing step that removes the noise, such as spell checking or text normalization [12–14]. Nonetheless, the trend nowadays is to use end-to-end approaches [15–17] that exploit the raw data from the source without applying preprocessing steps, in an attempt to harness every bit of information for the specific task at hand while also avoiding introducing early errors in the NLP pipeline. On the other hand, it is also not entirely clear whether a normalization approach outperforms the direct use of word embeddings on noisy texts [18]. Normalization, as a preprocessing step, will alter the original information encoded in the input text, although in a way that would benefit the next stages of the pipeline. For instance, if we normalize 'nooooo' to 'no', the emphasis of the first word is lost. In this case, we should highlight the *intentionality* when using one form over the other, which contrasts with accidentally introducing spelling mistakes in the writing which, nonetheless, may still convey some information such as the educational level of the writer. Granted, a system that only includes normalized words in its vocabulary will probably benefit from using the latter form instead.

In this work, we introduce an adaptation of the skipgram model from [10] to train word embeddings that better integrate word variants (otherwise considered noisy words) at training time. This can be regarded as an analogous incremental improvement over fastText to what this one was over word2vec. Then, we perform an evaluation on a wide array of intrinsic and extrinsic tasks, comparing their performance to that of well-known embedding models such as word2vec and fastText on both standard and noisy English texts. The results show a clear improvement over the baselines in semantic similarity and sentiment analysis tasks, with a general tendency to retain the performance of the best baseline on standard texts and outperform them on noisy texts. Our ultimate goal is to improve the performance of traditional embedding models in the context of noisy texts. This would alleviate the need for the usual preprocessing steps such as spell checking or microtext normalization, and act as a good starting point for modern end-to-end NLP approaches.

2. Towards Noise-Resistant Word Embeddings

Word embedding models such as word2vec, GloVe or fastText are able to cluster word variants together when given a big enough training corpus that includes standard and non-standard language [11]. That is, given enough examples where 'friend' (standard word), 'freind' (spell-checking error), 'frnd' (phonetic-compressed spelling) and even 'dog' or 'dawg' (street-talk) appear in similar contexts, these words will be translated to similar vector representations. Taking advantage of this fact, many state-of-the-art microtext normalization systems use word embeddings in their pipelines [14,19–21], both when generating normalization candidates for the input words and also when selecting them.

The problem with this approach is that the contexts where those example words appear are also likely to be affected by the same phenomena as the words themselves. For example, 'friend' might appear in phrases such as 'that's my best friend' or 'friend for life', while 'frnd' in others such as 'dats my bst frnd' or 'frnd 4 lifee'. This can make it difficult for the embedding algorithm to find the semantic similarity between 'friend' and 'frnd' when only relying on the assumption that the training corpus is big and diverse enough to effectively convey this variability. However, not all of the embedding algorithms are equally affected by this, as those which take subword information into account may have an advantage: in our example, the similar morphology shared by the word variants may be exploited by algorithms such as fastText, which uses character n-grams to give them more similar vector representations.

In this paper, we present a modification of the skipgram model proposed by Bojanowski et al. [10] (a modification of the original by Mikolov et al. [8]), which tries to improve the clustering of standard words and their noisy variants. This is attained through the use of *bridge-words*, normalized derivatives of the original words from the training corpus where one of their constituent characters is removed. By using these new words at training time in addition to the original ones, our objective is to increase the similarity between word variants, using those bridge-words as intermediate terms that match the words we want to cluster together. For example, 'friend' and 'freind' have in common the bridge-words 'frind' and 'frend'. Even if the original words do not appear in the same context in the training corpus, using the bridge-words in place of the originals allows for indirect paths to be discovered: 'friend'-'frind'-'freind' and 'friend'-'frend'-'freind'. In the case of 'friend' and 'frnd', and assuming that we use an embedding algorithm that exploits subword information, as we propose here, the higher morphological similarities of the latter with respect to the bridge-words 'frend' and 'frind' benefits their grouping together in the same cluster. As a side note, in the sense that these are intermediate (or normalized) representations that tie together otherwise isolated terms, they may resemble the index terms used in information retrieval. Since there is no index in our case, we will not refer to them as such. Notably, it should also be possible to apply analogous modifications to the ones described here to other training models, such as the continuous bag of words [8].

It is worth pointing out that we did not consider the latest state-of-the-art models such as BERT [22] and GPT-2 [23] as it would not be feasible to apply analogous modifications to these large

and complex models at this point, and GPT-3 [24] is out of the question in this regard. On the other hand, although we currently consider a monolingual English setup, our method should be suitable for any other language with a similar concept of *character*, in contrast to those based on logograms such as Chinese.

2.1. Modified Skipgram Model

The skipgram model found in tools like word2vec and fastText establishes that, for each word in a text, it should be possible to predict those in their corresponding contexts [8]. As a consequence, the words that appear in similar contexts end up represented by similar vectors, so that the transformation learned by the model can effectively map one group of words onto the other.

Based on the skipgram model from fastText, our proposal aims at increasing the similarity between standard words and their noisy counterparts. To accomplish this, we introduce a new set of words at training time that we denominate *bridge-words*. For each word in the training corpus, we first put the words into lowercase, strip the accents and remove character successive repetitions, and then obtain one bridge-word for each remaining character in the word, by removing one different character each time. For character repetitions, we cover both standard and non-standard ones (e.g., 'success' vs 'daaammn'), obtaining a common denominator for when users make the mistake of removing standard repetitions (e.g., from 'success' to 'succes'), or add repetitions to provide emphasis (e.g., from 'damn' to 'daaammn'). The resulting words are very similar and can still be read mostly in the same way. An analogous reasoning is used in the case of lowercasing and stripping the accents. Note that this procedure is exclusively applied to obtain all the bridge-words, and the unprocessed corpus will be used during training. Formally, let $\mathcal{V}$ be the word vocabulary extracted from the training corpus so that $\mathcal{V} = \{w_1, w_2, ..., w_n\}$ with n the size of the vocabulary. The set of bridge-words is then defined as $\mathcal{B} = \{b_{1,1}, b_{1,2}, ..., b_{1,|w_1|}, ..., b_{n,1}, b_{n,2}, ..., b_{n,|w_n|}\}$, where $|w_i|$ is the length of word w_i, and $b_{i,j}$ is the bridge-word obtained by first normalizing as described earlier and then removing the character at position j from the word $w_i \in \mathcal{V}$ (it is possible that $\mathcal{V} \cap \mathcal{B} \neq \emptyset$). These new words are used in addition to the original words when predicting their context in the skipgram model training, as depicted in Figure 1. For example, in the phrase 'that's my best Fríendd ever', the objective is not only to predict 'that's', 'my', 'best' and 'ever' using the word 'Fríendd', but also using the derived bridge-words 'riend', 'fiend', 'frend', 'frind', 'fried' and 'frien'. This idea of removing one character at a time is similar to the one used in the tool SymSpell (https://github.com/wolfgarbe/SymSpell) to speed up spell-checking, where it replaces the exhaustive approach of considering all possible edit operations (i.e., addition, removal, substitution and transposition).

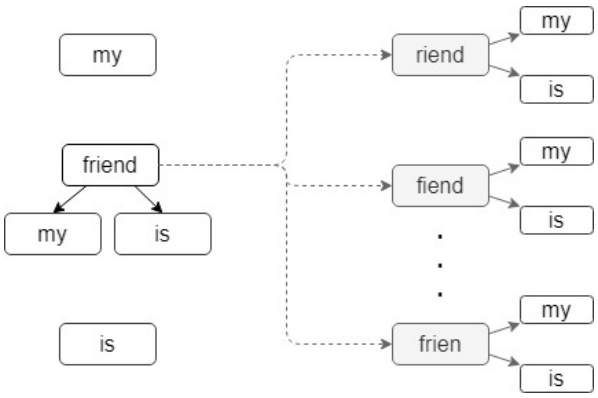

Figure 1. Visualization of the adapted skipgram model where we add bridge-words into its training.

In our case, bridge-words are not interesting *per se* but as *intermediaries* between other words. We do not require that they coincide with real words with which they would establish a direct connection; in fact, we assume that these connections will be indirect most of the time. For instance, we do not consider the substitution operations that would construct 'tome' and 'tame' from 'time',

which would explicitly connect the three, but only 'tme', which can be obtained from the three of them by removing one character, linking them together indirectly.

It is important to observe that these bridge-words also constitute artificial noise introduced in our training process that could play a harmful role. As an example, the word 'fiend' appears as a bridge-word for 'friend', while also being a standard word from the English dictionary without much semantic relation to the concept of friendship. Because of this, bridge-words should not have the same impact as the original words when tuning the parameters of the model. We propose two mechanisms for lowering the weight of bridge-words in the training process: (1) introducing them randomly, with a fixed probability p_b, instead of for all the original words, and (2) reducing the impact in the objective function by adding a weighting factor. Formally, let w_x be an input word of length $|w_x|$, b_j the bridge-word for w_x when the character at position j is removed, w_y a target word in the context of w_x, H a random variable with $P(H = 1) = p_b$ and $P(H = 0) = 1 - p_b$, $h \sim H$, λ the weight factor and $E_{ft}(w_x, w_y)$ the objective function of the skipgram model from fastText, then our new objective function, E_{robust} is defined as:

$$E_{robust} = E_{ft}(\mathbf{w_x}, \mathbf{w_y}) + h \cdot \lambda \cdot \sum_{j=1}^{|w_x|} E_{ft}(\mathbf{b_j}, \mathbf{w_y})$$

where $\mathbf{w_x}$, $\mathbf{w_y}$, and $\mathbf{b_j}$ are the vector representations of the corresponding input, target and bridge-words.

In any case, the proposed technique does not rule out the requirement of a training corpus where standard and noisy variants of words are used. Rather, it enhances the capacity of already existing models (in this case, the skipgram model from fastText) to *bridge* or further interconnect these word variants. The corresponding source code is available at https://github.com/yeraidm/bridge2vec.

3. Evaluation

We use multiple intrinsic and extrinsic evaluation tasks to study the performance of our approach together with word2vec and fastText. The models are trained using the same unprocessed corpus of web text and tweets. Starting with the usual word similarity task, we also include outlier detection [25], most of the extrinsic tasks from the SentEval benchmark [26], and then we add Twitter SA from various editions of the SemEval workshop. Ideally, we should see that our embeddings are able to retain the performance of 'vanilla' fastText embeddings [10] for standard and less-corrupted text, while outperforming them on noisier texts, and that word2vec [8] is at a disadvantage in this case.

It is worth noting that including models like BERT in our benchmarks would be unfair given their significantly higher complexity and the amount of resources employed in their training with respect to the current ones which, sadly, are out of our reach. Our initial aim was to improve upon existing easy-to-train models (much more affordable to researchers), leaving the latest language models out of the question for this work. After all, we are not presenting a totally new embedding model but a technique to enhance existing ones.

We believe that fastText and word2vec remain an accessible way to obtain competent embeddings in many scenarios, including for low-resource languages where bigger models would need more training data. In addition, this is not to mention the significantly higher amount of computational resources required by these models in general.

In any case, it would be interesting (although out of scope for this work) to confirm whether BERT and similar models work well on noisy texts, which would require testing them against comparable models that take this use-case more explicitly, as we do in our paper for smaller embedding models.

3.1. Word Embedding Training

In the present work, we use a combination of web corpora, specifically the UMBC corpus [27], and tweets collected through the Twitter Streaming API from dates between October 2015 and July

2018. It is worth noting that we did not perform any preprocessing or normalization step over the resulting corpus, and the final dataset is formed by 64.653 M lines and 3.3 B tokens, of which 24.558 M are unique.

We employed a modified version of the skipgram model from fastText which incorporates the changes described in Section 2.1 together with a vanilla version (https://github.com/facebookresearch/fastText) and a word2vec baseline (https://github.com/dav/word2vec), using the default hyperparameters for all models. In the case of the proposed model, we train four instances in order to take a first look at the influence of the hyperparameters introduced: the probability of introducing a bridge-word (p_b) and the weight for bridge-words in the objective (λ). The combinations are $(p_b = 1, \lambda = 1)$, $(p_b = 0.5, \lambda = 1)$, $(p_b = 1, \lambda = 0.1)$, and $(p_b = 0.5, \lambda = 0.1)$. In this work, we do not perform hyperparameter optimization given resource and time constraints, and those values were selected according to the initial hypothesis that a decreased impact of bridge-words in the training process should be beneficial to the model. The training of our models is four times slower than vanilla fastText and word2vec when $p_b = 0.5$ and 6.5 times slower when $p_b = 1$ on average.

3.2. Intrinsic Tasks: Word Similarity and Outlier Detection

The first intrinsic evaluation task is the well-known semantic word similarity task. It consists of scoring the similarity between pairs of words, and comparing it to a gold standard given by human annotators. In a word embedding space, the similarity between two words can be measured through a distance or similarity metric between the corresponding vectors in the space, such as cosine similarity. The evaluation is performed using the Spearman correlation between the list of similarity scores obtained and the gold standard. In this work, we use the wordsim353 [28], SCWS [29], SimLex999 [30], and SemEval17 (monolingual) [31] evaluation datasets.

The second task is outlier detection, which consists of identifying the word that does not belong in a group of words according to their pair-wise semantic similarities. As an example, snake would be an outlier in the set german shepherd, golden retriever, and french pitbull, in spite of also being an animal, since it is not a dog. In this case, we use the 8-8-8 [25] and wiki-sem-500 [32] datasets, and measure the proportion of times in which the outlier was successfully detected (i.e., the accuracy).

3.3. Extrinsic Tasks: The Senteval Benchmark and Twitter SA

Since it is not evident that performance on intrinsic tasks translates proportionally to extrinsic tasks [33,34], where word embeddings are used as part of bigger systems, we resort to the SentEval benchmark [26] in order to evaluate our embeddings in a more realistic setup. The tasks included in this benchmark evaluate sentence embeddings, which can be obtained from word embeddings using an aggregating function, which can go from the simple bag of words to the more complex neural-based models InferSent [35] or GenSen [36]. Additionally, some tasks require a classifier to be trained on the sentence embeddings in order to obtain an output of the desired type. In both cases, we maintain a simple approach where we focus on the raw performance of the word embeddings rather than the models used on top of them. This means using the bag of words model to obtain sentence representations, which simply averages the corresponding word embeddings from each sentence, and then linear regression for the classification tasks.

SentEval includes 17 extrinsic tasks, of which we use 16 and 10 probing tasks. The first group includes semantic textual similarity (STS 2012-2016, STS Benchmark and SICK-Relatedness), natural language inference (SICK-Entailment and SNLI), sentiment analysis (SST, both binary and fine-grained), opinion–polarity (MPQA), movie and product review (MR and CR), subjectivity status (SUBJ), question-type classification (TREC), and paraphrase detection (MRPC). The second group is formed by tasks that evaluate other linguistic properties which could be found encoded in sentence embeddings, such as sentence length, depth of the syntactic tree or the number of the subject of the main clause. For a more detailed description of these tasks together with references to the original sources, see [26]. In general, for the similarity tasks, the performance is measured using Spearman correlation, while,

in the rest of the cases, which correspond to classification tasks, the accuracy of the classification is obtained. Unfortunately, we leave image-caption retrieval task (COCO) out of our test bench as it is not possible to access the source texts. This would be needed for the processing that we perform as described in the next section.

Finally, we also evaluate on the SA datasets released in the SemEval workshops by Nakov et al. [37] (task 2, subtask B), Rosenthal et al. [38] (task 9, subtask B, using the training data from the previous edition), and Nakov et al. [39] (task 4, subtasks B, D, C, and E). These already include noisy texts in the form of tweets, thus they are not processed in the same way as the following datasets are processed, as explained below. However, since we still use the SentEval code, we did filter the neutral/objective tweets in ternary SA datasets. We also performed downsampling on the 2016 training and development datasets, both binary and fine-grained, in order to compensate for the substantial unbalance across instance classes. This is important as the test datasets are also skewed in the same manner, and it lead the classifiers to adjust to this bias to obtain unrealistic results. In the case of the binary task, we equated the positive instances with the number of negative ones, while, in the case of the fine-grained task, we used a fixed maximum number of 500 instances per class, given the huge gap between the least frequent class (accounting for 71 instances) and the most frequent one (including 2876 instances). Note that other datasets used in this work are also unbalanced, although to a significantly lesser extent and with no such measurable impact on the results.

3.4. Dataset De-Normalization

Since we could not find noisy text datasets for such a wide variety of evaluation tasks as the ones from the SentEval benchmark, we decided to *de-normalize* (i.e., introduce artificial noise into) these standard datasets, while also keeping the originals of the benchmark, in order to cover the case of noisy texts in the extension needed by this work. The procedure consists of randomly replacing every word in the texts with a noisy variant with some fixed probability. The noisy variants are obtained from two publicly available normalization dictionaries, `utdallas` and `unimelb`, released in the first edition (2015) of the W-NUT workshop [40], formed by (non-standard, standard) word pairs.

For the word similarity and outlier detection datasets, this probability p_d was fixed to 1; i.e., we modify all the words in the test set which appear in our normalization dictionaries (which cover 78.61% of them). In the case of the SentEval datasets, we created three versions for each one of them: a heavily corrupted version ($p_d = 1$), a more balanced version ($p_d = 0.6$), and a less noisy one ($p_d = 0.3$). As an example, from the original sentence 'A man is playing a flute,' we obtain 'aa woma isz playiin thw flute', 'aa mann is playng da flute', and 'aa wman is playing the flute', in each respective case. The Twitter SA datasets, on the other hand, were not de-normalized.

Furthermore, we perform ten de-normalization runs over the intrinsic tasks datasets and three over the extrinsic ones, obtaining multiple noisy versions of each dataset. By averaging the results over the different de-normalizations, we try to neutralize extreme measurements that can be caused by different noisy variants of words.

3.5. Results

Our currently best model is obtained with the hyperparameter combination $(p_b = 0.5, \lambda = 0.1)$, which in some way validates our hypothesis that bridge-words should be introduced in a restrained fashion. In general terms, this model has a similar performance to fastText in the standard case, while outperforming both word2vec and fastText in noisy setups, with wider margins towards noisier texts.

3.5.1. Intrinsic Evaluation

Table 1 shows the results on the intrinsic word similarity task. On standard words, fastText and our model obtain similar performance, both surpassing that of word2vec. On non-standard words, however, our model is able to consistently outperform fastText in every dataset, and word2vec

falls further behind possibly due to its lack of support for out-of-vocabulary words in this scenario, as 48.77% of the unique noisy test words are not included in the vocabulary of the word2vec model. Differences for non-standard words between our model and both word2vec and fastText are statistically significant under a significance level of 0.01.

Table 1. Spearman correlation results of word similarity on SCWS, wordsim353 (WS353), SimLex999 (SL999), and SemEval17 (Sem17) datasets.

	Standard			
	SCWS	**WS353**	**SL999**	**Sem17**
word2vec	64.7	69.1	32.2	68.2
fastText	**65.4**	72.7	33.5	70.3
ours	65.1	**73.1**	**33.8**	**70.4**
	Noisy			
	SCWS	**WS353**	**SL999**	**Sem17**
word2vec	13.0	13.7	−10.9	11.2
fastText	35.2	38.1	7.3	37.2
ours	**42.1**	**44.2**	**16.4**	**43.1**

In the case of outlier detection, shown in Table 2, we obtained mixed results and the differences between our model and the baselines are not statistically significant. On the 8-8-8 dataset, our model outperforms the baselines both in the standard and noisy scenarios, although with visibly lower margins than in the case of semantic similarity. However, on the wiki-sem-500 dataset, word2vec outperforms its competitors on standard words and does not lose much performance on the noisy setup. The latter may be explained by the low amount of successfully denormalized words, with just 7.5% of the total (compared to 52.2% on the 8-8-8 dataset), which also hints at the tie between fastText and our model.

Table 2. Accuracy results of outlier detection on 8-8-8 and wiki-sem-500 (wiki) datasets.

	Standard		**Noisy**	
	8-8-8	**wiki**	**8-8-8**	**wiki**
word2vec	59.4	**53.8**	22.8	39.3
fastText	65.6	49.0	31.7	**41.1**
ours	**67.2**	47.8	**33.3**	**41.1**

3.5.2. Extrinsic Evaluation

Given the considerable amount of tasks and datasets included in the SentEval benchmark, we decided to group similar tasks and datasets and show the aggregated results from each group instead of following an exhaustive approach. In this case, and given the variability in dataset sizes, we use a weighted average as the aggregation function.

First of all, we show in Figure 2 the dynamic behavior of each model when going from standard texts to noisier ones. In this case, we divided the tasks into two groups based on the performance metric: Spearman correlation or accuracy. The first one encompasses the semantic similarity and relatedness tasks (all STS* and SICK-Relatedness) and the second one the rest of them. Except in the case of word2vec on the first group (yellow lines and crosses), all the models start from a very similar position in the standard scenario. Then, the performance begins its downward trend, where our model starts to stand out above the baselines. As we go towards noisier texts, our model manages to stay above the rest of the lines, increasing the distance margin up until the last stretch.

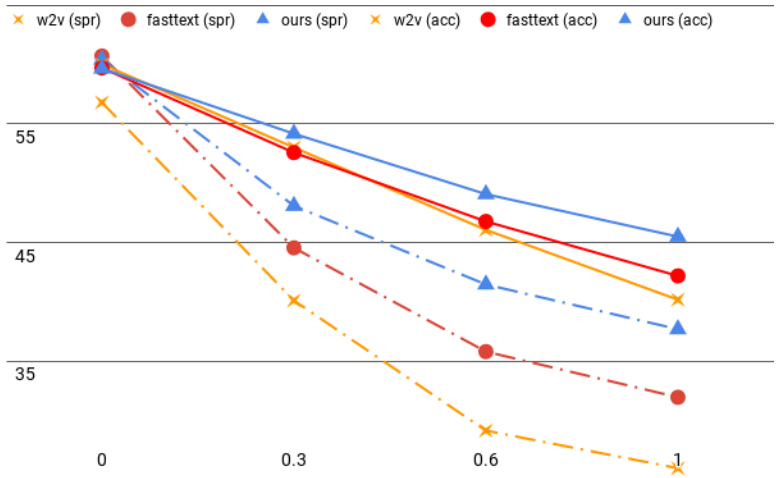

Figure 2. Performance of each considered model when going from standard texts to noisier ones on the extrinsic tasks. In lines and dots is the aggregated performance on semantic similarity and relatedness tasks (Spearman correlation). In continuous lines is the aggregated performance on the rest of the tasks (accuracy).

Next, Table 3 shows in greater detail the performance of each model in a less aggregated view. In this case, datasets have been grouped by task as described in Section 3.3. As we can see, our model is on par with the baselines on standard texts, with a few interesting exceptions: (1) it is able to obtain some advantage on sentiment analysis, which fastText also obtains over word2vec; (2) on question-type classification, word2vec obtains the best performance, and still clearly outperforms fastText on the lowest noise level, although not our model; and (3) on the probing tasks, word2vec takes the lead again, this time by a smaller margin. Regarding noisy texts, our model is clearly superior on semantic similarity and relatedness, as we had already seen before, and it also outperforms the baselines on the rest of the tasks, with wider margins on noisier texts, but with the sole exception of paraphrase detection. In this surprising case, word2vec outperforms both fastText and our model obtaining better accuracy on texts with the highest level of noise compared to the previous step. It appears that, with the proper training (and hence, vocabulary), word2vec remains a strong baseline on extrinsic tasks, even in the case of noisy texts, where the level of noise has to be increased notably in order for fastText to obtain a clear advantage. This can also be observed following the continuous lines in Figure 2. On the other hand, the weakness seen on word semantic similarity (Table 1) relating to out-of-vocabulary words does not seem to translate to extrinsic tasks, where having more context and hence a higher chance of finding in-vocabulary words mitigates the problem, as we can see in the semantic similarity and relatedness (Table 3) results. In any case, differences on noisy texts between our model and the baselines are statistically significant under a significance level of 0.05, with p-values below or barely above 0.01.

Finally, in Table 4, we show the results obtained on the SemEval Twitter SA datasets. In this case, word2vec continues to display a strong performance, fastText loses the advantage it had on the SentEval benchmark for the same SA task, and our approach is able to revert this performance loss to outperform, once again, both of the baselines. Performance differences are statistically significant under a significance level of 0.05, again with p-values below or barely above 0.01. At this point, we can observe how fastText is inferior to word2vec on a real-world social media setting, when we may have expected the opposite at first. However, for this same reason, it is remarkable to see our approach taking the lead despite being a modification of fastText, which also demonstrates the benefit of including the bridge-words at training time. Having said that, it would be relevant to investigate if higher performance figures can be obtained by modifying the skipgram model from word2vec.

Table 3. Results of the extrinsic evaluation on the SentEval benchmark. The noise levels are *low* ($p_d = 0.3$), *mid* ($p_d = 0.6$), and *high* ($p_d = 1$).

	Standard	Noisy			Standard	Noisy		
		Low	*Mid*	*High*		*Low*	*Mid*	*High*
	Semantic sim. & rel.				**Binary classification**			
word2vec	56.8	40.1	29.2	26.0	81.5	78.8	76.1	71.4
fastText	**60.7**	44.6	35.8	32.0	**81.8**	79.0	75.8	72.1
ours	60.4	**48.1**	**41.5**	**37.7**	81.6	**79.4**	**77.7**	**74.1**
	Sentiment analysis				**Entailment**			
word2vec	57.9	55.4	50.5	42.8	**66.3**	54.9	48.8	35.8
fastText	58.8	55.8	52.6	47.0	66.2	54.0	50.0	40.2
ours	**59.3**	**56.9**	**54.6**	**51.1**	**66.3**	**55.1**	**51.4**	**48.1**
	Question-type classification				**Paraphrase detection**			
word2vec	**79.4**	65.6	53.3	35.0	72.6	**67.0**	**61.2**	**65.5**
fastText	74.8	62.5	52.1	41.8	72.3	62.7	57.1	56.8
ours	73.4	**67.5**	**59.4**	**49.5**	**72.9**	66.9	60.2	56.3
	Probing tasks							
word2vec	**58.2**	51.5	45.3	39.3				
fastText	57.8	51.2	45.9	41.0				
ours	57.7	**52.8**	**48.4**	**43.6**				

Table 4. Accuracy results of the extrinsic evaluation on SemEval (SE) Twitter SA datasets.

	SE13 B	SE14 B	SE16 BD	SE16 CE
word2vec	84.3	88.3	77.4	35.1
fastText	83.3	88.1	76.5	33.7
ours	**84.8**	**88.6**	**78.4**	**35.5**

4. Related Work

Word embeddings have been at the forefront of NLP research for the past decade, although the first application of vector representation of words dates back to [41]. More recently, the first models to attain wide use were word2vec [8] and GloVe [9], which take words as basic and indivisible units, implying that the word vocabulary is fixed at training time and any unknown word would be given the same vector representation, regardless of its context or any other intrinsic property. To address the limitations of word2vec and GloVe with out-of-vocabulary words, where morphologically-rich languages such as Finnish or Turkish are specially affected, new models appeared which take subword information into account. The type of subword information used varies in each particular approach: some of them require a preprocessing step to extract morphemes [42], while others employ a less strict approach by directly using the characters [43,44] or character n-grams [10,45] that form the words.

When targeting noisy texts from social media, such as tweets from Twitter, previous work relies solely on the high coverage that can be obtained from training in an equally noisy domain [11]. An exception to this rule is the work from Malykh et al. [46], where they try to obtain robust embeddings to misspelled words (one or two edit operations away from the correct form) by using a new neural-based model. In this case, the flexibility is obtained by an encoding of the prefix, suffix and set of characters that form each word. By using this set of characters in the encoding, where the specific order between them is disregarded, this approach achieves some form of robustness to low-level noise, while the prefix and suffix part encodes most of the semantic information. The main difference of our approach is that we are not proposing a whole new model but a generic technique to adapt existing ones. This could be applied to many others, including that from Malykh et al. [46] itself.

Furthermore, we evaluate our embeddings in the context of non-standard texts, a noisier medium than the slightly misspelled standard texts regarded in [46]. Unfortunately, we could not include this approach in our test bench as, probably due to differences in the development environment setup, we were not able to train new models nor extract embeddings through pretrained models using the latest version of the code at https://gitlab.com/madrugado/robust-w2v/tree/py3_launch.

Lastly, if we consider standard and non-standard texts as pertaining to different languages, our approach would be similar to [47], where they also adapt the skipgram model to obtain bilingual embeddings. In this work, they start with comparable bilingual corpora and automatically calculate alignments between words across languages. At training time, they use the words from alignment pairs interchangeably in the texts from each language, requiring each word to predict not only the context in its own language but also the context in the other language. In our case, we only consider one training corpus and create a set of bridge-words that act as alignments between standard words and their noisy counterparts. On the other hand, the weight given to these new words in the objective function is $\lambda < 1$ as they represent noisy examples, whereas, in [47], the words from the other language are given more weight ($\lambda > 1$).

5. Conclusions

In this work, we have proposed a modification of the skipgram model from fastText intended to improve the performance of word embedding models on noisy texts as they are found on social media, while retaining the performance on standard texts. To do this, we introduce a new set of words in the training process, called bridge-words, whose objective is to connect standard words with their noisy counterparts.

We have evaluated the performance of the proposed approach together with word2vec and fastText baselines on a wide array of intrinsic and extrinsic tasks. The results show that, while the performance of our best model on standard texts is mostly preserved when compared to the baselines, it generally outperforms them on noisier texts with wider margins as the level of noise increases.

As future lines of research, we will perform the same study on other languages and adapt the proposed modification of the skipgram model to work with the newest BERT [22] and GPT-2 [23] models, given that GPT-3 [24] is prohibitively expensive to train. In light of its competitive performance, adapting the skipgram model from word2vec might prove useful. Other types of bridge-words such as phonetic codes obtained from a phonetic algorithm like the Metaphone [48] could also prove to be beneficial, in addition to a weighted term inversely proportional to word length, and u or inverted-u distributions for noise introduction. Additionally, our approach is orthogonal to other techniques that enhance the performance of word embeddings, such as the ones described in [49], and so they too can be applied to the models obtained in this work.

Author Contributions: Conceptualization, Y.D.; methodology, Y.D.; software, Y.D.; validation, Y.D. and J.V., and C.G.-R.; formal analysis, Y.D. and C.G.-R.; investigation, Y.D.; resources, Y.D.; data curation, Y.D.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D., J.V., and C.G.-R.; visualization, Y.D. and C.G.-R.; supervision, J.V. and C.G.-R.; project administration, J.V. and C.G.-R.; funding acquisition, J.V. and C.G.-R. All authors have read and agreed to the published version of the manuscript.

Funding: Yerai Doval has been supported by the Spanish Ministry of Economy, Industry and Competitiveness (MINECO) through the ANSWER-ASAP project (TIN2017-85160-C2-2-R); by the Spanish State Secretariat for Research, Development and Innovation (which belongs to MINECO) and the European Social Fund (ESF) through a FPI fellowship (BES-2015-073768) associated with TELEPARES project (FFI2014-51978-C2-1-R); and by the Xunta de Galicia through TELGALICIA research network (ED431D 2017/12). The work of Jesús Vilares and Carlos Gómez-Rodríguez has also been funded by MINECO through the ANSWER-ASAP project (TIN2017-85160-C2-1-R in this case); and by Xunta de Galicia through a Group with Potential for Growth grant (ED431B 2017/01), a Competitive Reference Group grant (ED431C 2020/11), and a Remarkable Research Centre grant for the CITIC research centre (ED431G/01), the latter co-funded by EU with ERDF funding. Finally, Carlos Gómez-Rodríguez has also received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, Grant No. 714150).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The Importance of Word Segmentation

In principle, when using word embeddings, we assume that the input text is correctly segmented into words. However, suppose that this is not the case, and that it goes beyond frequent de-normalization instances where words are joined or merged together; e.g., 'noway'-'no way', 'yesplease'-'yes please'; or even instances where the individual words cannot be immediately recovered such as with 'tryna'-'trying to', or 'whatchu'-'what are/do you'. Instead, in the present case, we will consider sentences like 'theproblem was veryclear' or 'the prob lem was very clear'. A possible solution would be to perform a word segmentation preprocessing step [50] before obtaining the corresponding word embeddings, which would imply introducing again the notion of sequential tasks together with the risk of error propagation.

However, let us now consider that the two operations involved in bad word segmentation (i.e., word joining and splitting) might not have the same impact on the process of obtaining relevant word embeddings. If we take into account that models such as fastText, and by extension the modification presented in this chapter, use subword information to construct word embeddings, we might argue that *joining words together* may be moderately supported by these models, as they would still consider the words inside the merging as character n-grams modelled during training. On the contrary, *splitting words* would be more problematic, as it removes parts of a word which could be crucial to obtain the adequate vector representation.

To check this hypothesis, we have devised new experiments using new de-normalized versions of the STS* datasets from the SentEval benchmark, which we have divided into two sets: *join* and *split*. In the former, we randomly removed word delimiters from input sentences with a fixed probability p_j, while in the latter we added delimiters between word characters with a lower probability p_s, $p_s < p_j$, in order to account for the higher amount of non-delimiter characters.

The results obtained, which are shown in Table A1, seem to support our hypothesis. Therefore, using a word segmenter with a slight tendency to join words (e.g., through a threshold parameter as shown by Doval et al. [51]) or even the raw input directly (taking into account the low frequency of splits, while joins are frequent in special elements such as hashtags or URLs), can be considered good practical solutions so long as we use embedding models that exploit subword information. Nonetheless, the latter option is especially relevant for us, since it shows that we may finally dispense with any form of input preprocessing for languages that delimit words; English in our current case. However, even in the case of Chinese, where words are not explicitly delimited and word segmentation is a well-studied and complex subject, it has been recently shown that this preprocessing step might not be necessary. Meng et al. [52] propose directly operating over Chinese characters rather than *strict* words. We highlight this strictness property as characters are frequently used as words themselves, but not always. That solution obtains better results than other systems that require a previous word segmentation step, even when all of them are implemented as state-of-the-art neural networks. In our case, this shows that we could relax the definition of a *word*, and obtain the embeddings at the character- or sequence-of-characters level.

Table A1. Spearman correlation averages on the new de-normalized STS* datasets, with $p_j = 0.5$ and $p_s = 0.1$.

	$\textbf{Join}_\rho$	$\textbf{Split}_\rho$
word2vec	11.0	18.3
fastText	**39.2**	**18.7**
ours	**39.2**	17.2

References

1. Bansal, M.; Gimpel, K.; Livescu, K. Tailoring Continuous Word Representations for Dependency Parsing. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; Association for Computational Linguistics: Baltimore, MD, USA, 2014; pp. 809–815. [CrossRef]
2. Vulić, I.; Moens, M.F. Monolingual and Cross-Lingual Information Retrieval Models Based on (Bilingual) Word Embeddings. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval—SIGIR '15, Santiago, Chile, 11–15 August 2015; pp. 363–372. [CrossRef]
3. Kutuzov, A.; Velldal, E.; Øvrelid, L. Redefining part-of-speech classes with distributional semantic models. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 11–12 August 2016; Association for Computational Linguistics: Berlin, Germany, 2016; pp. 115–125. [CrossRef]
4. Xiong, S.; Lv, H.; Zhao, W.; Ji, D. Towards Twitter sentiment classification by multi-level sentiment-enriched word embeddings. *Neurocomputing* **2018**, *275*, 2459–2466. [CrossRef]
5. Lampos, V.; Zou, B.; Cox, I.J. Enhancing Feature Selection Using Word Embeddings: The Case of Flu Surveillance. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 May 2017; ACM: Perth, Australia, 2017; pp. 695–704. [CrossRef]
6. Yang, X.; Macdonald, C.; Ounis, I. Using word embeddings in Twitter election classification. *Inf. Retr. J.* **2018**, *21*, 183–207. [CrossRef]
7. Liang, S.; Zhang, X.; Ren, Z.; Kanoulas, E. Dynamic Embeddings for User Profiling in Twitter. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD, London, UK, 19–23 August 2018; ACM: London, UK, 2018; pp. 1764–1773. [CrossRef]
8. Mikolov, T.; Corrado, G.; Chen, K.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the 1st International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013; pp. 1–12. [CrossRef]
9. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1532–1543. [CrossRef]
10. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
11. Sumbler, P.; Viereckel, N.; Afsarmanesh, N.; Karlgren, J. Handling Noise in Distributional Semantic Models for Large Scale Text Analytics and Media Monitoring. In Proceedings of the Abstract in the Fourth Workshop on Noisy User—Generated Text (W-NUT 2018), Brussels, Belgium, 1 November 2018.
12. Eisenstein, J. What to do about bad language on the internet. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–14 June 2013; Association for Computational Linguistics: Atlanta, GA, USA, 2013; pp. 359–369.
13. Chrupała, G. Normalizing tweets with edit scripts and recurrent neural embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; Association for Computational Linguistics: Baltimore, MD, USA, 2014; pp. 680–686. [CrossRef]
14. Van der Goot, R.; van Noord, G. MoNoise: Modeling Noise Using a Modular Normalization System. *Comput. Linguist. Neth. J.* **2017**, *7*, 129–144.
15. Bordes, A.; Boureau, Y.; Weston, J. Learning End-to-End Goal-Oriented Dialog. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
16. Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; Rush, A. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In Proceedings of ACL 2017, System Demonstrations, Vancouver, BC, Canada, 30 July–4 August 2017; Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 67–72.
17. Schmitt, M.; Steinheber, S.; Schreiber, K.; Roth, B. Joint Aspect and Polarity Classification for Aspect-based Sentiment Analysis with End-to-End Neural Networks. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 1109–1114.

18. van der Goot, R.; Plank, B.; Nissim, M. To normalize, or not to normalize: The impact of normalization on Part-of-Speech tagging. In Proceedings of the 3rd Workshop on Noisy User-generated Text, Copenhagen, Denmark, 7 September 2017; pp. 31–39.
19. Costa Bertaglia, T.F.; Volpe Nunes, M.D.G. Exploring Word Embeddings for Unsupervised Textual User-Generated Content Normalization. In Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT), Osaka, Japan, 19 November 2020; The COLING 2016 Organizing Committee: Osaka, Japan, 2016; pp. 112–120.
20. Ansari, S.A.; Zafar, U.; Karim, A. Improving Text Normalization by Optimizing Nearest Neighbor Matching. *arXiv* **2017**, arXiv:1712.09518.
21. Sridhar, V.K.R. Unsupervised text normalization using distributed representations of words and phrases. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, Denver, CO, USA, 31 May–5 June 2015; pp. 8–16.
22. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. 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: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019; Burstein, J., Doran, C., Solorio, T., Eds.; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; Volume 1 (Long and Short Papers), pp. 4171–4186. [CrossRef]
23. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
24. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
25. Camacho-Collados, J.; Navigli, R. Find the word that does not belong: A Framework for an Intrinsic Evaluation of Word Vector Representations. In Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, Berlin, Germany, 12 August 2016; Association for Computational Linguistics: Berlin, Germany, 2016; pp. 43–50. [CrossRef]
26. Conneau, A.; Kiela, D. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018; European Language Resources Association (ELRA): Miyazaki, Japan, 2018.
27. Han, L.; Kashyap, A.L.; Finin, T.; Mayfield, J.; Weese, J. UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*; Association for Computational Linguistics: Atlanta, GA, USA, 2013; pp. 44–52.
28. Finkelstein, L.; Evgeniy, G.; Yossi, M.; Ehud, R.; Zach, S.; Gadi, W.; Eytan, R. Placing Search in Context: The Concept Revisited. *ACM Trans. Inf. Syst.* **2002**, *20*, 116–131.
29. Huang, E.; Socher, R.; Manning, C.; Ng, A. Improving Word Representations via Global Context and Multiple Word Prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Jeju Island, Korea, 8–14 July 2012; Association for Computational Linguistics: Jeju Island, Korea, 2012; pp. 873–882.
30. Hill, F.; Reichart, R.; Korhonen, A. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Comput. Linguist.* **2015**, *41*, 665–695. [CrossRef]
31. Camacho-Collados, J.; Pilehvar, M.T.; Collier, N.; Navigli, R. SemEval-2017 Task 2: Multilingual and Cross-lingual Semantic Word Similarity. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, BC, Canada, 3–4 August 2017; Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 15–26. [CrossRef]
32. Blair, P.; Merhav, Y.; Barry, J. Automated Generation of Multilingual Clusters for the Evaluation of Distributed Representations. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.

33. Faruqui, M.; Tsvetkov, Y.; Rastogi, P.; Dyer, C. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, Berlin, Germany, 12 August 2016; Association for Computational Linguistics: Berlin, Germany, 2016; pp. 30–35. [CrossRef]
34. Chiu, B.; Korhonen, A.; Pyysalo, S. Intrinsic Evaluation of Word Vectors Fails to Predict Extrinsic Performance. In Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, Berlin, Germany, 12 August 2016; Association for Computational Linguistics: Berlin, Germany, 2016; pp. 1–6. [CrossRef]
35. Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; Bordes, A. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; Association for Computational Linguistics: Copenhagen, Denmark, 2017; pp. 670–680.
36. Subramanian, S.; Trischler, A.; Bengio, Y.; Pal, C.J. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018; pp. 1–16.
37. Nakov, P.; Rosenthal, S.; Kozareva, Z.; Stoyanov, V.; Ritter, A.; Wilson, T. SemEval-2013 Task 2: Sentiment Analysis in Twitter. In Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, GA, USA, 14–15 June 2013; Association for Computational Linguistics: Atlanta, GA, USA, 2013; pp. 312–320.
38. Rosenthal, S.; Ritter, A.; Nakov, P.; Stoyanov, V. SemEval-2014 Task 9: Sentiment Analysis in Twitter. In Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, 23–24 August 2014; pp. 73–80.
39. Nakov, P.; Ritter, A.; Rosenthal, S.; Sebastiani, F.; Stoyanov, V. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 1–18. [CrossRef]
40. Baldwin, T.; de Marneffe, M.C.; Han, B.; Kim, Y.B.; Ritter, A.; Xu, W. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In Proceedings of the Workshop on Noisy User-—Generated Text, Beijing, China, 31 July 2015; pp. 126–135.
41. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533. [CrossRef]
42. Luong, T.; Socher, R.; Manning, C. Better Word Representations with Recursive Neural Networks for Morphology. In Proceedings of the 17th Conference on Computational Natural Language Learning, Sofia, Bulgaria, 8–9 August 2013; Association for Computational Linguistics: Sofia, Bulgaria, 2013; pp. 104–113.
43. Ling, W.; Dyer, C.; Black, A.W.; Trancoso, I.; Fermandez, R.; Amir, S.; Marujo, L.; Luís, T. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 5 June 2015; Association for Computational Linguistics: Lisbon, Portugal, 2015; pp. 1520–1530. [CrossRef]
44. Kim, Y.; Jernite, Y.; Sontag, D.A.; Rush, A.M. Character-Aware Neural Language Models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; AAAI Press: Phoenix, AZ, USA, 2016; pp. 2741–2749.
45. Wieting, J.; Bansal, M.; Gimpel, K.; Livescu, K. Charagram: Embedding Words and Sentences via Character n-grams. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–4 November 2016; Association for Computational Linguistics: Austin, TX, USA, 2016; pp. 1504–1515. [CrossRef]
46. Malykh, V.; Logacheva, V.; Khakhulin, T. Robust Word Vectors: Context-Informed Embeddings for Noisy Texts. In Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User—Generated Text, Brussels, Belgium, 9–15 September 2018; pp. 54–63.
47. Luong, T.; Pham, H.; Manning, C.D. Bilingual Word Representations with Monolingual Quality in Mind. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, Denver, CO, USA, 5 June 2015; Association for Computational Linguistics: Denver, CO, USA, 2015; pp. 151–159. [CrossRef]
48. Philips, L. Hanging on the metaphone. *Comput. Sci.* **1990**, *7*, 39–43.

49. Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; Joulin, A. Advances in Pre-Training Distributed Word Representations. In Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018; European Language Resources Association (ELRA): Miyazaki, Japan, 2018.
50. Doval, Y.; Gómez-Rodríguez, C. Comparing neural-and N-gram-based language models for word segmentation. *J. Assoc. Inf. Sci. Technol.* **2019**, *70*, 187–197. [CrossRef] [PubMed]
51. Doval, Y.; Gómez-Rodríguez, C.; Vilares, J. Spanish word segmentation through neural language models. *Proces. Del Leng. Nat.* **2016**, *57*, 75–82.
52. Meng, Y.; Li, X.; Sun, X.; Han, Q.; Yuan, A.; Li, J. Is Word Segmentation Necessary for Deep Learning of Chinese Representations? In Proceedings of the 57th Conference of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 3242–3252.