Enhancing Subword Embeddings with Open N-grams

Csaba Veres(✉) and Paul Kapustin

University of Bergen, Bergen, Norway
{csaba.veres,pavlo.kapustin}@uib.no

Abstract. Using subword n-grams for training word embeddings makes it possible to subsequently compute vectors for rare and misspelled words. However, we argue that the subword vector qualities can be degraded for words which have a high orthographic neighbourhood; a property of words that has been extensively studied in the Psycholinguistic literature. Empirical findings about lexical neighbourhood effects constrain models of human word encoding, which must also be consistent with what we know about neurophysiological mechanisms in the visual word recognition system. We suggest that the constraints learned from humans provide novel insights to subword encoding schemes. This paper shows that vectors trained with subword properties informed by psycholinguistic evidence are superior to those trained with ad hoc n-grams. It is argued that physiological mechanisms for reading are key factors in the observed distribution of written word forms, and should therefore inform our choice of word encoding.

1 Introduction

There is currently a great deal of research activity around solutions using continuous representations of words. The most popular methods for learning word vectors, or *embeddings*, produce a single vector for each word form in the training set, for example GloVe [18], word2vec [15], and SVD [12]. These methods do not attempt to exploit syntactic or morphological regularities behind the word forms, as the unit of analysis is the single word.

These methods could be regarded as modern day experiments inspired by Zellig Harris' hypotheses about the distributional structure of language. Harris proposed that word meanings give rise to observable distributional patterns in language, such that two semantically unrelated words A and C would be less likely to be found in common linguistic contexts as two semantically related words A and B [10]. Modern machine learning techniques have made it computationally possible to *embed* very high dimensional distributional patterns in a much lower dimensional vector space, in which the distances between any given vectors are related to the similarities of context in which the corresponding words are found in the training set. Semantic relatedness is therefore correlated with the calculated distance (e.g. cosine distance) between vectors.

E. Métais et al. (Eds.): NLDB 2020, LNCS 12089, pp. 3–15, 2020.
https://doi.org/10.1007/978-3-030-51310-8_1

Since the unit of analysis in such machine learning models is the word, it is generally the case that rare words, and words formed by novel morphological combinations, are not represented in the training set with sufficient frequency to obtain a vector embedding. In response, Bojanowski et al. [2] trained vectors by decomposing words into subword components. Their embeddings had the advantage that low frequency and out-of-vocabulary words could be assigned a vector representation from the sum of their subword units. The training model is a modification of the skipgram model [15], in introducing a scoring function over the sum of the component n-grams (including the word itself). A word is therefore represented as the sum of the vector representation of the word and its constituent n-grams. Conversely, the vector representation of individual n-grams is shared between all words containing the particular n-gram, and therefore rare words can acquire reliable representations by taking advantage of the shared representations.

While the reported evaluations of the embedding vectors look promising, our experience in using them in an application has been mixed. Using the standard implementation[1] *fastText* and our own vectors trained with the latest Wikipedia dumps[2], we observed some examples where the related words would not be particularly useful for some tasks, for example query expansion. Consider the fairly common word *dictionary*, which has the following nearest vectors: *dictionaries, dictionarys, dictionarian, dictionarie, dictions, dictionarial, dictioneer, dictioner, dictionaric, dictionay.* These are highly overlapping morphological variations and not particularly useful in applications where more heterogeneous semantically related concepts are required for information retrieval. In contrast, word2vec[3] provided the following results for this example: *dictionaries, lexicon, oed, merriam, encyclopedia, bibliographical, lexicography, britannica, websters, encyclopaedia, glossary.* In general we had the intuition that fastText was more likely to include overlapping orthographic clusters in the results set, which motivated the experiments reported in this paper. We wanted to understand why, and under what circumstances the *fastText* results might suffer, and developed the hypothesis that psycholinguistic factors were involved. The approach is similar in spirit to emerging efforts which explore modern machine learning results from a psycholinguistic perspective. For example, Mandera et al. [14] use *semantic priming* results from psycholinguisic experiments instead of semantic similarity judgements to evaluate word vectors, and report new insights in the quality of embeddings.

The subword embedding approach presupposes that orthography to semantics mappings can be established for words as well as for the summation of the subword fragments. Thus, the vector obtained for out-of-vocabulary items acquires its position in semantic 'co-occurance' space as a function of the

[1] https://github.com/facebookresearch/fastText.

[2] We also used publicly available pretrained vectors, e.g. wiki-en.bin https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.en.zip but found these even less satisfactory.

[3] https://github.com/dav/word2vec.

semantic space vectors in the constituent n-grams. Consider the following example adapted from [2], where the word *where* contains the following set of *tri*-grams. Note that the word is actually represented by n-grams where n is greater than or equal to 3 and less than or equal to 6, but for the sake of brevity we only list the trigrams:

<wh, whe, her, ere, re>

where the symbols ' $<$ ' and ' $>$ ' denote word boundaries.

Each of these trigrams are shared by many other words with very different meanings, for example:

$< wh$ appears in $which, whence, whale, white, whack, whack-a-mole,$ $wharf, whatever, ..$
whe appears in $anywhere, arrowhead, wheel, wheat, ...$
her appears in $whether, cherish, butcher, sheriff, thermometer, ...$

It seems clear that vectors generated from short n-grams will be at a point in semantic space that is not necessarily close to the semantics of any of the words which contain them, because they are contained in many words. The longer the n-gram, the fewer the containing words. It might seem odd that the inclusion of short n-grams in training would do any good at all, because they appear to introduce semantic noise. In fact, a systematic evaluation of n-gram length shows that embeddings that include bi- and tri- grams always show a slight degradation in performance when compared to those with only longer n-grams [2].

An additional consideration about the number of words sharing subword elements originates in psycholinguistic evidence about the mental representation of words. An important variable in human word recognition studies is the *orthographic neighbourhood*, commonly measured with Coltheart's N [3], where the orthographic neighbourhood of a word is determined by counting the number of other words of the same length, sharing all but one letter, and maintaining letter position. The measure of orthographic neighbourhood is correlated with the number of words related to the target word, which overlap significantly with the *set of* subword components. Every n-gram in a word is shared by a variety of semantically unrelated words, but the set of the constituent n-grams is unique to a particular word. Or, to put it in the opposite way, each word contributes its context to the training of its unique set of n-grams. When this set is re-assembled, the summed vector will be consistent with the vector just for the word itself. But this will be maximally effective when the combined set of constituent n-grams does not overlap significantly with another set of n-grams corresponding to a different word with a different meaning.

For example, the word *safety* has the *tri*-grams

<sa, saf, afe, fet, ety, ty>

and there are no other six-letter words which have a significant overlap with this set. On the other hand the word *singer* has the *tri*-grams

<si, sin, ing, nge, ger, er>

which overlap significantly with many semantically unrelated words such as *sinner, finger, linger*

<si, sin, inn, nne, ner, er>, <fi, fin, ing, nge, ger, er>, <li, lin, ing, nge, ger, er>

The result of this overlap is that if we present a rare word which overlaps significantly with these n-grams, its vector representation will be degraded to the extent that the set of overlapping n-grams is shared by semantically unrelated words. For example the semantically unrelated word *zinger* (a striking or amusing remark) will have similar sets of n-grams:

<zi, zin, ing, nge, ger, er>

which has a 67% overlap with *finger* and *linger*. The assembled vector for "zinger" would therefore be somewhere between "finger" and "linger".

The set of overlapping words in these examples is just what we have called the orthographic neighbourhood. In the previous example, this means that the trigrams for the high-N six-letter words have a 50%–67% overlap with N other words in its lexical neighbourhood, whereas for the low-N it is none, except for morphological variants of the word itself[4]. The higher the N, the more likely it is that a significant subset of n-grams will be shared by semantically unrelated words.

This paper explores the hypothesis that orthographic neighbourhood structure of English has some influence on the way subword n-grams are incorporated into word embeddings. We first describe some relevant findings involving orthographic neighbourhoods that have come to light as a result of psycholinguistic theories. We then show empirically how these properties can influence the quality of word embeddings, and propose alternative encoding schemes inspired by psycholinguistics and neuroscience, which solve some of the problems. Our main contribution is to show that a consideration of words as more than letter strings in some disembodied vocabulary, is beneficial. Words are the result of psycholinguistic processes. Based on this argument we develop a putatively better encoding scheme which takes into consideration the interdependency between word structure and human psychological and neural processing systems.

2 Orthographic Neighbourhood Density

Coltheart's N is the simplest measure of orthographic neighbourhood density, where two words are neighbours if they have the same length and differ in just one letter position. There have been many refinements, including the counting of words formed by letter transposition or repetition, and the use of Levenshtein distance. Nevertheless, many of the fundamental results hold for neighbourhoods with the Colthert's N measure, which we use in this paper [1,3,7,22].

The neighbourhood density of words is correlated with their length. [1] counted the number of neighbours for four, five, and six letter words in the

[4] In this example, *safe* has overlapping n-grams with *safety*.

CELEX linguistic database. In total these amounted to 8956 words, and she found a systematic difference; four-letter words had on average 7.2 neighbours, five-letter words had 2.4, and six-letter words, 1.1 neighbour. Not surprisingly, longer words tend to have fewer neighbours. Experiments which specifically manipulate neighbourhood density tend to use shorter words, typically four-letter words.

Orthographic neighbourhood N has an effect on how quickly people can respond to stimuli in a lexical decision task (LDT). In LDT, subjects are presented with a random sequence of words and nonwords, and asked to classify them as quickly as possible. Coltheart et al. [3] found that high-N nonwords (nonwords with many word neighbours) were classified more slowly than nonwords with few word neighbours. That is, people would be slower to respond that *dinger* was not a word than that *rafety* was not. This result is consistent with our view that nonword letter sequences that are similar to many words will be subject to more interference from existing word representations.

The effect of N on the word stimuli in LDT is less clear, but the preponderance of evidence suggests that words with high neighbourhoods are classified faster than words with low neighbourhoods. Thus, while having lots of neighbours hinders nonword classification, it helps word classification. Large lexical neighbourhood also helps in the naming task, where subjects are required to pronounce words presented to them. Andrews [1] reviewed eight studies and found that they all showed facilitatory effects. On the face of it, these findings appear to contradict the hypothesis that high neighbourhood words should have lower quality representations. However, one problem with interpreting the psycholinguistics evidence is that the results might not bear directly on the quality of the representations but, rather, on the decision process employed by the subjects in the experiment. That is, if a stimulus can generate lots of potential word vectors then a decision process might be more ready to accept it as a word - which is helpful if in fact it is a word, but unhelpful if it is not. The reaction time would then be influenced by the number of vectors rather than their quality.

However, a more intriguing possibility is that the human word recognition system constructs representations in such a way that high-N words are not disadvantaged by their overlapping lexical neighbours. If it is true that machine learning techniques can suffer in high-N environments but humans do not, then it would be advantageous to learn from the human system. We therefore decided to find more concrete evidence about the effects of neighbourhood density on the quality of trained embeddings.

2.1 Experiment 1: Effects of Orthographic Neighbourhood on Word Embeddings

Perhaps it goes without saying that there are currently no tests of word embeddings which take orthographic neighbourhood density into consideration. As a first step we decided to do a post-hoc analysis on popular data sets which are used for evaluating embeddings: SimLex-99 [11], WS353 [5], and the Stanford

Rare Word (RW) Similarity Dataset [13]. We counted the neighbourhood density of every unique word in each data set, as reported in Table 1. The average densities were surprisingly high; for example Forster and Shen [6] limit their high neighbourhood condition to $N \geq 4$, and other experimenters typically consider 4–5 as being high-N. Table 1 also shows the distribution of words of various lengths in the dataset, as well as the weighted mean length. WS353 has a slightly higher distribution of longer words and a correspondingly lower neighbourhood than SimLex, but most interestingly the RW set showed quite a different distribution with many more long words and a corresponding decrease in the N. We take this to be a completely uncontrolled and accidental feature of the datasets.

The differences in neighbourhoods suggest an alternative explanation for results obtained by [2], who found that English word embeddings from word2vec performed slightly better than fastText in the WS353 test set, but fastText performed better with RW. Their explanation was that the words in WS353 are common words for which good word2vec vectors can be obtained without exploiting subword information. The words in the RW dataset, on the other hand, are helped by subword information because their whole-word vectors are poor. Our analysis of neighbourhoods suggests a quite different explanation. By our hypothesis, fastText embeddings perform best in low-N words environments, which is only the case for the RW data set.

Encouraged by this evidence, we devised a more direct plausibility test for our hypothesis, further inspired by the observation that many of the high-N words we entered into the fastText nearest neighbour[5] query tool returned results where many of the words were morphologically related. They seemed to retain a core morphological stem. For example the high-N query word *tone* has the following semantic neighbours: *tones, overtone, staccato, toned, overtones, dissonantly, accentuation, accentuations, intonation, intonations*. One possible explanation for the morphological overlap is that a critical n-gram such as *ton* becomes central to the representation because it has to be intact to capture the semantic meaning. That is, *tone* has 20 orthographic neighbours, **one: bone done gone lone none cone hone pone sone zone t*ne: tine tune tyne to*e: toke tole tome tope tore tote ton*: tons* and disrupting the morpheme *ton* gives a completely different word. Interestingly, this phenomenon seems to extend to morphemes that are not part of the original query word. For example *bone* has fastText semantic neighbours: *bones, tooth, cartilage, marrow, teeth, osteo, arthroplastie, osteoarthritic, osteochondral, osteolysis* and word2vec neighbours: *bones, cartilage, tooth, skull, marrow, tissue, femur, fractures, teeth, spine*. Again, fastText appears to have latched on to an orthographic cluster which has stable semantic context, whereas word2vec has a much more varied answer set. We wanted to quantify this observation, and the test we proposed was to count the number of unique word stems returned for a nearest neighbour query. That is, by using a common stemming algorithm, we were able to eliminate results which shared

[5] It is unfortunate that words with similar embeddings are sometimes called 'neighbour', e.g. on http://fasttext.cc. To avoid confusion we will refer to these as 'semantic neighbours' from now on.

Table 1. Percentage of words of length l in three common data sets, the mean length of words in the data set and their observed Coltheart's N

Word Length	SimLex-99	WS353	RW
1	0	0.2	0
2	2	0.2	0.1
3	8	5.5	1.45
4	18	13	4.8
5	20	14.7	6.2
6	19	20	6.7
7	13.7	11.9	11.7
8	7	10	14.1
9	6.2	9	14.8
10	3.4	6	12.7
11	1.7	4.6	11
12	0.5	2	7.5
13	0.01	0.9	4.4
14	0.01	0.4	2.7
15			0.88
16			0.5
17		0.27	
18			0.03
19			0.03
Mean length	5.8	6.63	8.78
N	4.83	3.45	1.47

a common stem. There are several commonly used stemming algorithms [17], and none of them necessarily eliminate all the orthographically derived forms we would like, but after some experimentation we used the popular Snowball stemmer from NLTK.

We trained a word2vec and a fastText model on the latest WikiPedia data dump as of March 2018[6] (enwiki 20180320). All word2vec and fastText models were trained on the same dump to ensure consistency. The build of word2vec was obtained from https://github.com/dav/word2vec and fastText from https://github.com/facebookresearch/fastText.git.

A set of 9 low-N and 9 high-N, 4-letter words were assembled, keeping word length constant. These were submitted to word2vec and fastText to compute the top 10 nearest semantic neighbours. Each word was stemmed with the Snowball stemmer in the NLTK toolbox[7]. Finally the number of unique stems was subtracted from the total number of semantic neighbours for the two

[6] https://dumps.wikimedia.org/.
[7] http://www.nltk.org/howto/stem.html.

conditions. Table 2 shows the number of words that share some unique stem. fastText performed about as well as word2vec on low-N words, but seemed to suffer on the high-N words, corroborating our intuition that fastText vectors were sub optimal with high-N words.

Table 2. Number of words sharing a common stem

	Low-N	High-N
word2vec	14	15
fastText	12	20

3 Models of Word Encoding

Experiment 1 gave some reason to believe that the simple n-gram model of subword representation might be limiting the quality of the vectors obtained with fastText, because of the presence of high-N words. Since orthographic neighbourhoods effects are predominantly psycholinguistic, we reasoned that drawing on existing knowledge about human word encoding schemes, might help us to improve orthographic word representations. Our hypothesis is that the surface form of writing systems evolved in light of the properties of the human orthographic encoding mechanism, and an understanding of the properties of the human encoding system could help us implement coding schemes which are better suited to process those surface forms. The Latin alphabet provides discrete components which are combined to form words of various lengths. Interestingly, even though short words tend to have higher neighbourhood densities, there is an inverse relation between word length and frequency of use in English function words [16]. That is, the most frequently used words tend to be short with potentially high orthographic neighbourhoods, which could lead to errors if the perceptual system was not adapted to avoid the errors. Psycholinguistic results about neighbourhood density effects form a key source of evidence for models of visual word recognition.

There is general consensus in the literature that abstract letter identities, independent of type font and case, are involved in the initial stages of printed word recognition [7]. Beyond that, there are differing proposals for how the letter detectors combine to enable printed word recognition. It is clear, for example, that letter position must somehow be computed because readers have no trouble distinguishing between, say, *bale* and *able*. On the other hand humans seem to be unperturbed by letter transposition, deletion, or insertion, such that the intended meanings of *tmie*, *grdn*, and *garxden* are generally recognised [9].

One of the most well supported proposals for an orthographic encoding schema is the *open bigram* (or more generally open n-gram) model of spatial encoding. In the open bigram model, letter encoding includes distant bigram pairs in words, not just spatially contiguous ones. For example, the word *clam* could be coded by the set {*cl, ca, cm, la, lm, am*}. Here the character gap between the letters is not constrained.

Whitney [21] proposed an interesting version of the open bigram approach in the SERIOL model, which incorporates facts about the role of neural activation in information representation. The model postulates a *letter level* encoding stage in which the relative position of each letter in the word string is encoded by the temporal neural firing patterns. That is, the subsequent *bigram level* recognises ordered pairs of letters, converting the temporal sequence into a spatial one. Bigram nodes recognize ordered pairs of letters and fire only if, for example, input A is followed by input B but not if only A were received, or if B were received prior to A. The neuronal firing occurs if letter nodes transmit their information within an oscillatory cycle, so non contiguous letter pairs can also activate letter bigrams, but the strength of firing is diminished with the distance between characters.

The bigram encoding model has similarities with the model of subword encoding in fastText. There are also important differences, in that Whitney's model uses only bigrams as well as non-contiguous bigrams. We decided to try if the introduction of non-contiguous/open n-grams, in analogy with the human perceptual system, could improve the performance of fastText embeddings.

3.1 Experiment 2: Non-contiguous n-grams

In this experiment we tested the addition of open n-grams to subword features, to see if they improved fastText vector representations. We experimented many different encoding schemes, and found that including both open, and regular contiguous n-grams gave inferior results to just using open n-grams[8]. In other words, contiguous n-grams always degraded performance. The best results were obtained by 300 dimensional vectors trained with n-grams where $3 \leq n \leq 6$. We call the trained models with only open n-grams *fasterText*, because every word has fewer n-gram components, and the model is slightly faster to train[9].

To illustrate the reduction in the number of components compared to contiguous n-grams, consider just the *tri*-grams for the word *safety* from our previous example, showing a 66% reduction in just the number of *tri*-grams:

contiguous trigrams: <sa saf afe fet ety ty>
open trigrams: <a e s f t a e y f t>

The performance of the fasterText vectors is shown in Table 3 for the previously described tests in Table 1, as well as the SemEval2012 task 2, and the Google analogy task. The former of these adds some fine grained semantic relations, and the latter some syntactic as well as semantic analogies. The results

[8] This is an interesting result since it departs from theories of human representation. We return to this point in the discussion.

[9] For example time to train the two best performing models on an Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30 GHz, 40 cores, 62 GB RAM, fastText real time = 376 m36.311 s, fasterText real time = 361 m35.879 s.

Table 3. Correlation between human judgement and word vector similarity scores on SimLex999 (semantic similarity, **not** relatedness), WS353 (semantic similarity and relatedness combined), SemEval2012 task 2 (various complex semantic relations), Google (semnatic and syntactic analogy task), and RareWord dataset (semantic similarity). Non-contiguous n-grams in fasterText shown against word2vec and fastText. Five different hyperparamters in fasterText are shown, where the number in parentheses is the degree of n. o+c also includes closed bigrams.

	SimLex	WS353	SemEval	Google	RW
word2vec	.33	.64	**.18**	**.71**	.41
fastText	.33	.69	.17	.69	.44
fasterText(2o+c)	.33	.66	.17	.68	.38
fasterText(2)	.33	.67	.17	.69	.39
fasterText(2–3)	.33	.68	**.18**	.7	.4
fasterText(2–6)	.33	.69	**.18**	.69	.42
fasterText(3–6)	**.34**	**.70**	**.18**	**.71**	**.45**

show Spearman's rank correlation coefficient between vector cosine distance and human judgement. fasterText embeddings achieve the best result (or equal best) on all of the five tests. This in spite of the fewer total n-gram components. The table also shows that performance tended to increase as longer n-grams were included, and degraded if bigrams were also present. However, the important point again is that the open bigram trained embeddings outperformed or equalled the state-of-the-art algorithms on every test.

Table 4. Results from *jiant* target tasks.

	mnli (accuracy)	kerte (accuracy)	sts-b (spearman r)	wnli (accuracy)
fastText (pretrained)	0.408	0.552	0.218	0.563
fastText	0.408	0.552	0.244	0.563
fasterText	**0.440**	**0.578**	**0.248**	0.563

3.2 Downstream Tasks

We compared the embeddings on several downstream tasks using the *jiant*[10] toolkit to evaluate on several GLUE[11] benchmarks [20].

The network configuration, including pretrain tasks, was taken from the *jiant* tutorial. The core model is a shared BiLSTM and no effort was made to optimize the model, since we were looking for a comparison between the embeddings

[10] https://jiant.info/.
[11] https://gluebenchmark.com/.

rather than state-of-the-art performance. The test tasks were the Multi-Genre Natural Language Inference (MultiNLI) for textual entailment, the Recognizing Textual Entailment (RTE), the Semantic Textual Similarity Benchmark (sts-b) and the Winograd Natural Language Inference (WNLI or also known as The Winograd Schema Challenge), from the GLUE test set. These tasks are all putatively semantic, indicating semantic encoding in the word vectors [20]. Table 4 shows the results of the comparison between pretrained fastText vectors obtained through the *jiant* web site, the fastText vectors trained on our corpus and the fasterText vectors trained on the same corpus. There is a slight improvement with fasterText in these downstream tasks, suggesting that these embeddings encode more precise semantic information.

In a separate line of work, HaCohen-kerner et al. [8] used Skip Char Ngrams and other character level features in stance classification of tweets. Their goal was to maximise the available features for the short texts available in individual tweets, and to reduce the effect of noisy data due to misspellings. They generated a large number of skip character features, skipping the range between 2–6 characters depending on the word, and found that their skip character ngrams outperformed previous benchmarks. While this work did not use word embeddings directly, it nevertheless shows that non contiguous n-grams provide unique information about word meanings.[12]

4 Summary, Conclusions, and Future Work

The use of subword information for training word embeddings benefits rare words, and languages where words are spontaneously derived by compounding. It was argued that subword encoding is also intrinsic to human word recognition, and the experiments showed that by including aspects of the known human encoding scheme in machine learning, the results can be improved across the board.

One curious aspect of the results was that regular, closed n-grams tended to reduce the quality of the embedding vectors. This is unusual because all psychological models we are aware of include regular n-grams. One possible explanation is that our model misrepresented the role of closed n-grams because it used only a simple model of open n-grams. In our implementation we put equal weight on each n-gram, irrespective of its serial position in the word. In addition, we only used a gap of one character between letters. This corresponds most closely with a *discrete open bigram* model, where non contiguous bigrams within a specified inter-letter distance receive an activation of 1, all other bigrams 0. Other approaches allow larger gaps and weighting functions, which result in different contributions of the subword units. For example the unit *cm* is activated in the word *clam* if two intervening letters are permitted, but not by the word *claim*. On the other hand the *continuous open bigram* model assigns a continuous and decreasing activation to bigrams that are separated by more letters. Thus

[12] We would like to thank an anonymous reviewer for bringing our attention to this work.

cm would receive some activation from both words, but more from *clam*. An obvious next step is to implement a version of the continuous model. This could be achieved by repeating bigrams by a factor that is proportional to the distance between them. That is, spatially closer n-grams would get more training cycles in any given word. By doing this we might be able to re introduce shorter n-grams which would improve out of vocabulary performance as well as retain its other good characteristics.

Hannagan et al. [9] argue that orthographic encoding can be mathematically modelled as a string kernel, and different encoding schemes are simply different parameters of the kernel. String kernels are a general approach originally designed for protein function prediction, and are consistent with a general, biologically plausible model of sequence comparison that is tolerant of global displacement and local changes. Our main contribution is to show that subword embeddings based on biologically plausible string kernels produce better results than embeddings based on ad hoc letter combinations. The claims should therefore apply also to other languages and writing scripts, as well as to other methods for generating embedding vectors, for example BERT [4] and ELMo [19]. Observed word forms evolve in conjunction with the capabilities and properties of the human visual system. Encoding schemes used in artificial neural networks could benefit from learning about the properties of real neural networks and the environments in which they operate.

References

1. Andrews, S.: The effect of orthographic similarity on lexical retrieval: resolving neighborhood conflicts. Psychon. Bull. Rev. **4**(4), 439–461 (1997). https://doi.org/10.3758/bf03214334
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv:1607.04606 (2016)
3. Coltheart, M., Davelaar, E., Jonasson, J.T., fir Besner, D.: Access to the internal lexicon. Attent. Perform. **VI**, 535–555 (1977)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
5. Finkelstein, L., et al.: Placing search in context: the concept revisited. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001, pp. 406–414. ACM, New York (2001). https://doi.org/10.1145/371920.372094
6. Forster, K.I., Shen, D.: No enemies in the neighborhood: absence of inhibitory neighborhood effects in lexical decision and semantic categorization. J. Exp. Psychol. Learn. Mem. Cogn. **22**(3), 696 (1996). https://doi.org/10.1037/0278-7393.22.3.696
7. Grainger, J., van Heuven, W.: Modeling letter position coding in printed word perception. In: Bonin, P. (ed.) The Mental Lexicon (2004)
8. HaCohen-kerner, Y., Ido, Z., Ya'akobov, R.: Stance classification of tweets using skip char ngrams. In: Altun, Y., et al. (eds.) Machine Learning and Knowledge Discovery in Databases, vol. 10536, pp. 266–278. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71273-4_22

9. Hannagan, T., Grainger, J.: Protein analysis meets visual word recognition: a case for string kernels in the brain. Cogn. Sci. **36**, 575–606 (2012). https://doi.org/10.1111/j.1551-6709.2012.01236.x
10. Harris, Z.S.: Distributional structure. WORD **10**(2–3), 146–162 (1954). https://doi.org/10.1080/00437956.1954.11659520
11. Hill, F., Reichart, R., Korhonen, A.: Simlex-999: evaluating semantic models with (genuine) similarity estimation. arXiv:1408.3456 (2014)
12. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. Trans. Assoc. Comput. Linguist. **3**, 211–225 (2015). https://scholar.google.com/scholar?cluster=6441605521554204538
13. Luong, M.T., Socher, R., Manning, C.D.: Better word representations with recursive neural networks for morphology. In: CoNLL, Sofia, Bulgaria (2013)
14. Mandera, P., Keuleers, E., Brysbaert, M.: Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: a review and empirical validation. J. Mem. Lang. **92**, 57–78 (2017). https://doi.org/10.1016/j.jml.2016.04.001
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
16. Miller, G., Newman, E., Friedman, E.: Length-frequency statistics for written English. Inf. Control **1**, 370–389 (1958). https://doi.org/10.1016/s0019-9958(58)90229-8
17. Moral, C., de Antonio, A., Imbert, R., Ramírez, J.: A survey of stemming algorithms in information retrieval. Inf. Res. Int. Electr. J. **19**(1) (2014). Paper 605
18. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014). http://www.aclweb.org/anthology/D14-1162
19. Peters, M.E., et al.: Deep contextualized word representations. arXiv:1802.05365 (2018)
20. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: a multitask benchmark and analysis platform for natural language understanding. In: The Proceedings of ICLR (2019)
21. Whitney, C.: How the brain encodes the order of letters in a printed word: the seriol model and selective literature review. Psychon. Bull. Rev. **8**(2), 221–243 (2001). https://doi.org/10.3758/BF03196158
22. Yarkoni, T., Balota, D., Yap, M.: Moving beyond Coltheart's N: a new measure of orthographic similarity. Psychon. Bull. Rev. **15**, 971–979 (2008)