ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

Word Embeddings for Morphologically Rich Languages

Pyry Takala *

Aalto University - Department of Computer Science
PL11000, 00076 Aalto - Finland

Abstract. Word-embedding models commonly treat words as unique symbols, for which a lower-dimensional embedding can be looked up. These representations generalize poorly with morphologically rich languages, as vectors for all possible inflections cannot be stored, and words with the same stem do not share a similar representation. We study alternative representations for words, including one subword-model and two character-based models. Our methods outperform classical word embeddings for a morphologically rich language, Finnish, on tasks requiring sophisticated understanding of grammar and context. Our embeddings are easier to implement than previously proposed methods, and can be used to form word-representations for any common language processing tasks.

1 Introduction

Most natural language processing tasks start by manipulating word, which raises the questions: what are words, and how should they be represented? Recently, words have been represented as multi-dimensional embeddings where between-word commonalities can be inferred from the vector-representations. This representation has helped practitioners to get excellent results in word-labeling [1] and other language processing tasks.

While embedding models have been successful with languages such as English, challenges arise with languages where words have multiple surface forms. Out-of-vocabulary rates become unacceptably high, and word-based language modelling fails for many inflectional languages [2]. Long language-processing pipelines [3] are often required, and tasks such as translation are challenging [4].

To embed words, a number of different alternatives have been developed.

Word-based embedding models consider unique strings as individual tokens. To create a lookup-table of word-embeddings, one can, for instance, feed a neural network with a "one-hot" vector that contains one dimension for each unique word, and the network can be trained on some relevant tasks. The weights attached to the input can be interpreted as the vector representation.

Subword-based models can use a similar approach as word-based models, where representations for sub-parts are looked up, and then combined. This approach can be competitive to word-based models, and can require significantly less parameters. This approach have been used for infrequent words [2], to find representations for discontinuous linguistic units [5], and some have also turned

*The author wishes to acknowledge Aalto Science-IT project and CSC - IT Center for Science, Finland, for computational resources.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

to morphemes as the basic unit [6]. Challenges include the decisions: how do we break the word, and how do we then combine subparts.

Character-by-character transformations start with the basic units of language, and combine these into more complex representations (e.g. morphemes). A character-based model could work with any language, discovering the language structure without human intervention. First character-based models have been applied succesfully to language processing tasks recently [7, 8].

Ideally, a language model would discover invariant units of language independently, without complex processing pipelines and training procedures. In this paper, we introduce an introduce an easy-to-use *Stem + ending* representation and we show that a character-level model can rival traditional word-embedding models for languages with rich morphology. The resulting embeddings can be used in common language processing tasks.

2 Proposed Embedding Methods

In this section, we describe the embedding models that we have experimented with. The models are illustrated in Figure 1 [1].

Regular embedding. Words are represented as vectors that have one dimension for each possible word, with only one active dimension. We limit the dictionary size to 50k and change other words to token "RAREWORD".

Regular, stemmed. As regular embedding, but with each word stemmed.

Stem + ending. A subword model where we break the words into a stem vector (40k dimensional), concatenated with a vector representing the remaining part (10k dimensional). We choose this alternative for simplicity: accessing a standard stemmer is easier than, for instance, using a morphological analyzer or training a recurrent neural network.

Moving-average. A character-based word-representation would ideally have small dimensionality, but receive information from all parts of the word. For languages that inflect at the end, the ending should be strongly present in the embedding. This motivates us to design a fixed transformation that is essentially a moving-average calculation. We create a vector, with the length of possible characters (100 after preprocessing). Starting with the first character in a word, we choose its corresponding location in the vector and increment this value with a value proportional to the character's position in the word, counting from the beginning. We get a word-representation $\mathbf{w} = (w_a\, w_b\, \ldots\, w_z\, \ldots)^T$,

where

$$w_a = \sum \frac{(1-\alpha)^{c_a}}{Z},$$

where c is the index of the character at hand (first character=0, second character=1, etc.), α denotes a hyper-parameter controlling the decay, and Z is a normalizer proportional to word length. This gives larger weights to characters

[1] Several other models could have been tested but have been omitted from this research. Other interesting representations could include a morpheme-based subword-model, and various complex character-based transformations, such as convolutional and recurrent neural networks.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

at the beginning of a word, and smaller weights to later characters. We also concatenate to the vector this transformation backwards. and add a vector of character counts, resulting to a dimensionality of 300.

Regular + moving-average. Regular concatenated with moving-average.

Learnt character-based. To test how well a learnt character-by-character transformation works, aiming for simplicity, we implement an MLP network. This network works similarly to the word-embedding, but on a character-by-character level. Each character is converted to a vector with the dimensionality of possible characters. The resulting character-vectors are concatenated to form a word-vector. We pad short words and cut longer words than 25 characters, resulting to dimensionality of 2.5k.

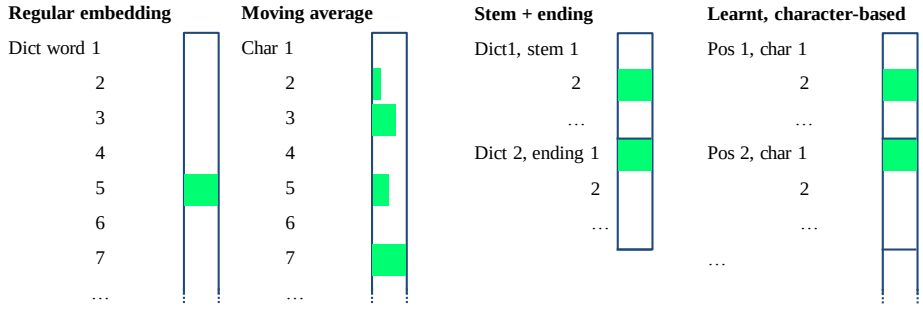

Fig. 1: Input Alternatives

3 Experiment Tasks and Data

Collobert et al. [1] use unsupervised learning and learn useful representations for a number of linguistic tasks. Taking inspiration from this setup, our task is a binary prediction for a word sequence on whether this sequence is real text (1), or grammatically or semantically questionable (0). To generate examples for this task, we choose from each sentence a window of seven words [2].

In the *Random Word Task*, we generate the randomly chosen negative examples by changing the middle word to a word drawn randomly from the set of all possible words, with the probability of each word proportional to its occurrence in the training set. A (stemmed) example is shown in Figure 2. In the *Inflected Word Task*, we choose the negative examples by changing a word with a word that has the same stem, but a different surface form. For instance, "dog" could be replaced with "dogs".

For our experiments with English text, we download the Simple English Wikipedia (1.2m sentences). For morphologically rich text, we use Selkosanomat, a newspaper written in simple Finnish (80k sentences) and the Finnish Wikipedia (1m sentences). We do some preprocessing steps, shuffle our sentences and

[2] In early experiments, we get best accuracies with window-size seven. Our choice of neural network architecture requires fixed word-windows, so we pad shorter lines.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

Original	There is a great deal of variance in color ...
Pos. example	there is a great deal of varianc
Neg. example	there is a politician deal of varianc

Fig. 2: Example of *Random Word Task*

pick word-windows uniformly at random from sentences. The data is split into training, validation and test, errors being reported on the test set.

4 Experiments

To keep our analysis focused on the input representation, we use a simple neural network model (see Figure 3.). We feed pre-processed word-windows to a network that starts by converting each word to a vector-representation, using one of the transformations described in the previous section. These word-vectors are fed to a word-specific MLP-layer, where each vector is processed (MLPs at different sentence position share here same weights). We concatenate the resulting vectors into one word-window vector, feed this to another MLP-layer, and a Softmax classifies the text as real or random.

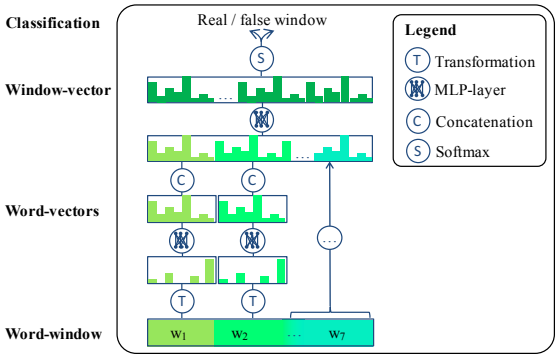

Fig. 3: Network Architecture

Before experiments, we optimize the network's hyper-parameters [3] for accuracy using a smaller English dataset of simple English.

In all our tasks, the problem is to distinguish a false word-window from a real one, with baseline accuracy of 50%. Our results are summarized in tables 1 and 2. For most interesting results, we perform training five times with different random seeds, showing average accuracies and standard deviations in brackets [4].

[3] We use one word-specific neural layer (250 neurons) and one word-window-specific layer (250 neurons), both with rectified linear units, and train for 20 epochs.

[4] To avoid penalizing overfitting experiments, we show best end-of-epoch accuracy. Some learnt character-based experiments omitted as results were poor already with small dataset.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Table 1: Random Word Task Results

	Finnish Wiki	**Selkosanomat**	**Simple-EN Wiki**
Regular	0.204	0.281 (0.028)	**0.142**
Regular, stemmed	0.225	0.247 (0.070)	0.166
Stem + ending	**0.142** (0.020)	**0.104** (0.042)	0.188
Moving-average	0.244	0.250	0.174
Regular + moving-average	0.232	0.244	0.208
Learnt character-based	na	0.302	0.268

Table 2: Inflected Word Task Results

	Finnish Wiki	**Selkosanomat**
Regular	0.268 (0.012)	0.260 (0.024)
Regular, stemmed	0.5*	0.5*
Stem + ending	0.244 (0.019)	0.269 (0.022)
Moving-average	**0.230** (0.023)	0.266 (0.022)
Regular + moving-average	0.244	**0.232**
Learnt character-based	na	0.352

* Not sensible: correct and random examples get same form after stemming.

Our embeddings are clearly advantageous for Finnish datasets. The models of *Stem + ending*, *Moving-average* and *Regular + moving-average* each outperform regular word-embeddings in different tasks. *Stem + ending* model outperforms the regular embedding, showing that splitting words of morphologically rich languages into subparts can have significant advantages. The performance of character-based models is also interesting, as character-based models have often not been able to rival word-based models, and as they can use a significantly smaller number of parameters ($5 * 10^5$ instead of 10^7).

A visualization of word endings (Figure 4) indicates that the vector representations of word endings are highly meaningful. Endings *ella*, *ellä*, *llä* and *öllä* tend to be all variations of a certain inflection, relate to being *at* something. Endings with *a* replaced with *ä* tend to also always represent the same meaning, only in the context of different word. The *istä* and *asta* translate both to *from*.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

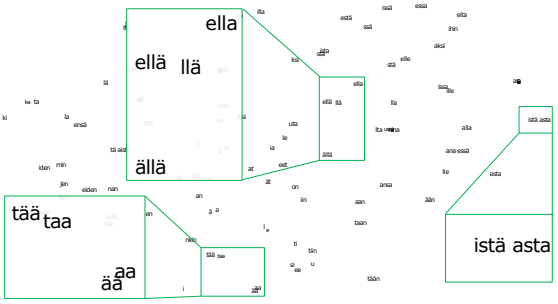

Fig. 4: Ending Vectors, Projected with t-SNE

5 Conclusion

Our work shows that subword and character-based models can rival regular word-embeddings for a morphologically rich language. *Stem + ending* vectors improve results for many language processing tasks and are very easy to implement. Also character-based models achieve good results, outperforming regular embeddings on certain tasks. Constructing a character-level neural network that automatically discovers meaningful units appears to be a promising research direction, and recent work parallel to ours (e.g. [9]) has started exploring this approach.

References

[1] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, (12), 2011.

[2] Tomás Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and J. Cernocky. Subword language modeling with neural networks. *Preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 2012.

[3] Jenna Kanerva, Juhani Luotolahti, Veronika Laippala, and Filip Ginter. Syntactic N-gram Collection from a Large-Scale Corpus of Internet Finnish. In *Proceedings of the Sixth International Conference Baltic HLT*, 2014.

[4] Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. Translating into Morphologically Rich Languages with Synthetic Phrases. *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.*, 2013.

[5] Wenpeng Yin and Hinrich Schutze. Deep Learning Embeddings for Discontinuous Linguistic Units. *arXiv preprint arXiv:1312.5129*, 2013.

[6] Jan A. Botha and Phil Blunsom. Compositional morphology for word representations and language modelling. *arXiv preprint arXiv:1405.4273*, 2014.

[7] Xiang Zhang and Yann LeCun. Text Understanding from Scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[8] Eric. Malmi, Pyry. Takala, Hannu. Toivonen, Tapani. Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. *arXiv preprint arXiv:1505.04771*, 2015.

[9] Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. *arXiv preprint arXiv:1508.00657*, August 2015.