9th International Conference on Speech Prosody 2018
13-16 June 2018, Poznań, Poland

Effects of Word Embeddings on Neural Network-based Pitch Accent Detection

Sabrina Stehwien, Ngoc Thang Vu, Antje Schweitzer

University of Stuttgart, Germany
{sabrina.stehwien,thang.vu,antje.schweitzer}@ims.uni-stuttgart.de

Abstract

Pitch accent detection often makes use of both acoustic and lexical features based on the fact that pitch accents tend to correlate with certain words. In this paper, we extend a pitch accent detector that involves a convolutional neural network to include word embeddings, which are state-of-the-art vector representations of words. We examine the effect these features have on within-corpus and cross-corpus experiments on three English datasets. The results show that while word embeddings can improve the performance in corpus-dependent experiments, they also have the potential to make generalization to unseen data more challenging.

Index Terms: pitch accents, convolutional neural networks, word embeddings

1. Introduction

Automatically detecting pitch accents from transcribed speech is a well-established field of research [1, 2, 3]. Most approaches use supervised statistical learning methods on time-aligned data and focus on labelling either words or syllables. Best results are obtained when using good-quality speech corpora that have been manually transcribed and prosodically annotated. The acoustic features are fairly standardized, usually consisting of pitch and energy values extracted per frame or aggregated across segments. Most methods benefit from the addition of lexical features such as part-of-speech tags [4, 5, 6, 7, 8]. Studies on the relationship between text and prosody in automatic detection methods dates back to early research in the field [9, 2]. Research on cross-genre prominence detection has also taken textual features into account [10, 11]. The prediction of prosodic prominence from text has been widely investigated for text-to-speech (TTS) synthesis; for example, it has been shown that certain words are accented quite frequently, and that this has an effect on prediction performance [12].

In a previous study we have introduced a model that uses a convolutional neural network (CNN) as a simple and efficient way of detecting pitch accents [13]. This method has the advantage that very little preprocessing is necessary due to the fact that the input data consists of low-level acoustic features. In line with the paradigm of using information that is as low-level as possible and letting the model learn feature representations of its own, we test an extension of this model that includes word embeddings [14, 15] as lexical information. Word embeddings are vector representations of words that are a standard method of encoding text in state-of-the art neural network models. Trained on large amounts of data in an unsupervised fashion, neural network algorithms "embed" words into vector space according to their textual context. Word embeddings have shown to encode both syntactic and semantic similarity [16]. While research in neural-network based TTS synthesis has made use of word embeddings to predict prominence from text [17, 18], to the best of our knowledge this paper is the first to test word embeddings for pitch accent detection on transcribed speech. We examine the effect of extending our acoustic model with word embeddings to within-corpus and cross-corpus experiments on three English corpora that are manually annotated with reference pitch accents. Our results show that adding word embeddings increases pitch accent detection performance on within-corpus experiments. In cross-corpus experiments, it is found that word embeddings are strong features that have the potential to overfit, thus making generalization more challenging.

2. Datasets

All datasets used in this work have been previously used for pitch accent detection and have been manually annotated with ToBI labels [19]. For our experiments, in which we aim to classify words as being pitch accented or not, the ToBI labels are grouped together into two classes: *accented* and *none*. Unsure events, that is events where the annotator was not certain of the presence of a pitch accent, are grouped into the *none* class. Table 1 lists the number of words and accented words in each dataset used in our experiments.

The first datasets is a subset of the Boston University Radio News Corpus (BURNC) [20] that was recorded from 3 female and 2 male speakers. The Boston Directions Corpus (BDC) [21] contains monologues by 4 speakers (3 male, 1 female) giving directions, both as read and spontaneous speech. The third corpus is the LeaP corpus [22], collected for prosodic research on non-native speech. In our experiments we use part of the LeaP corpus that consists of read stories and story retellings by non-native speakers with different L1 backgrounds. The subset contains speech recorded from 35 different speakers (22 female, 8 male) across varying proficiency levels. In total the subset comprises 14 different L1 backgrounds. Previous work [23] has shown that despite the lower proficiency, the automatic pitch accent detection performance on LeaP is comparable to that on native speech.

Table 1: *Dataset statistics*

corpus	words	accented	majority class
BURNC	26742	13780	51.5% accented
BDC	19225	8646	55.0% accented
LeaP	14651	6340	56.7% none

3. Baseline Acoustic Model

The following section describes the supervised learning model that is trained to label each word as carrying a pitch accent or not. A schematic overview of the baseline architecture is shown in the left part of Figure 1.

10.21437/SpeechProsody.2018-146

3.1. CNN-based model

The baseline model consists of a 2-layer convolutional neural network (CNN), following the model previously introduced for this task [13]. The CNN takes as input frame-based acoustic features representing the current word and its right and left direct context words. The signal is divided into s overlapping frames and represented by a d-dimensional feature vector for each frame. Thus, for each word, a matrix $W \in R^{d \times s}$ is formed as input. The first convolution layer consists of 100 2-dimensional kernels of the shape $6 \times d$ and a stride of 4×1, with d as the number of features. The kernels encompass the whole feature set to ensure that all features are learnt simultaneously. The second layer consists of 100 kernels of the shape 4×1 and a stride of 2×1. After applying two convolution layers, max pooling is used to find the most salient features. The max pooling size is set so that the output of each max pooling on each of the 100 feature maps has the shape $x = 100$. Finally, we apply dropout [24] with $p = 0.2$, which means that 20% of the neurons are corrupted in order to avoid overfitting. We also use $l2$ regularization. The resulting feature representation is fed into the softmax layer that consists of 2 units to form a binary classifier.

3.2. Acoustic features

The acoustic features are extracted from the speech signal using the openSMILE toolkit [25]. The features are 6 low-level acoustic descriptors of pitch, intensity and amplitude as provided in the openSMILE default feature sets: root mean square signal frame energy, loudness, smoothed F0, voicing probability of the F0 candidate, harmonics-to-noise ratio and zero-crossing rate. The intensity (energy, loudness) features are computed for each 20 ms frame and the remaining features for each 50 ms frame. All are computed with a 10ms shift.[1] The features do not require speaker-normalization [13]. The time intervals that indicate the word boundaries provided in the corpus are used to create the input feature matrices by grouping all frames for each word into one matrix. Afterwards, zero padding is added to the end of the input matrix to ensure that all matrices have the same size.

Including the context words in the feature matrix introduces information that does not pertain to the current word. For this reason, it was found [13] that the model greatly benefits from position features (or indicators) that are appended as an extra feature to the input matrices (see Figure 1). These features indicate the parts of the matrix that represent the current word. The rest of the matrix consists of zeros in this dimension. In the first convolution layer we ensure that the kernels always include the position indicator, hereby keeping the model informed which frames belong to the current word.

4. Lexico-Acoustic Model

4.1. Model extension

The lexico-acoustic model is an extension of the baseline acoustic model. The extension consists of an additional feed-forward network that learns features from word embeddings, shown in Figure 1 (in green shading). The input consists of one word embedding vector per input word. The vector values are used as non-trainable matrix weights. This is fed into the hidden layer, which has only a few output units n that form a "bottleneck".

[1]This frame size combination proved best in preliminary experiments.

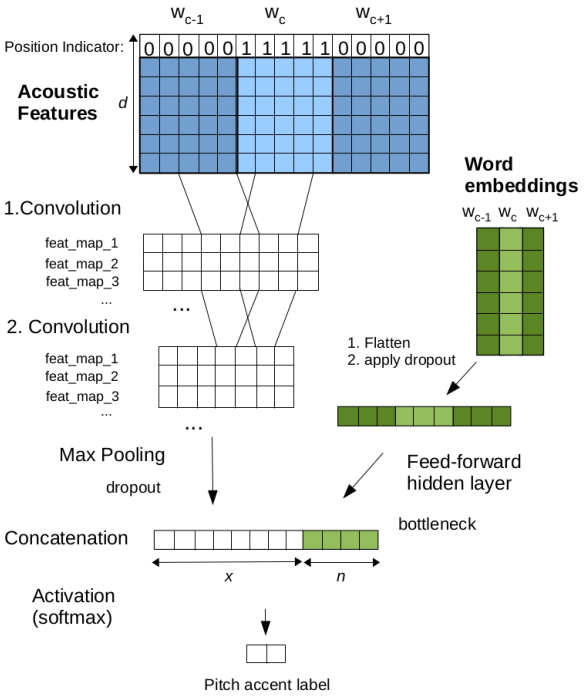

Figure 1: *Model overview for pitch accent detection using a CNN for acoustic features and extended by adding a feed-forward network for word embeddings, using an input context window of 3 words.*

Dropout with $p = 0.8$ is applied to the input before feeding it through the hidden layer. Afterwards, the two output layers of the acoustical and lexical models are regularized separately before they are concatenated into one final feature representation. This is fed into the softmax classification layer.

4.2. Word embeddings

As input features, we test two different types of word embeddings: GloVe [14] and word2vec [15]. The respective algorithms are usually applied to very large datasets to train the embeddings. Since the datasets used in this paper are comparatively small, we use the pre-trained 300-dimensional versions available online. These embeddings are used to represent all words in the datasets. Both GloVe and word2vec yield a number of out-of-vocabulary (OOV) words. We removed noise from the word labels such as special characters and we split contractions, e.g. *she'll* $\rightarrow$ *she* and hyphenated words, e.g. *eighty-eight* $\rightarrow$ *eight*. In both cases we kept the part of the word expected to be relevant for the presence or absence of a pitch accent. This way, the OOV rate was reduced to the numbers shown in Table 2. We set the embedding vectors for OOV words to consist of ones.[2] The word2vec pre-trained embeddings do not include the frequent words *a, and, of* and *to*, referred to as stopwords. We take this difference in to account in Table 2. These stopwords make up most of the OOV words in the word2vec embeddings, and are very rarely accented. The remaining OOV words, however, are frequently accented. The effect of stopwords represented as OOV in word2vec is discussed in section 5.2.4.

As input information we use unigram and trigram word em-

[2]This way we ensure that OOV embeddings receive a separate representation without having to custom-train them.

Table 2: *Out-of-vocabulary words for both word embedding types on all three datasets.*

	BURNC	BDC	LeaP
GloVe OOV			
tokens	233	19	4
types	64	11	4
accent rate	93%	74%	50%
w2v OOV			
tokens	3375	2496	1822
types	231	66	6
stopword rate	70.5%	87%	99.9%
accented stopwords	3%	13%	6%
accented remaining	79.5%	83%	100%

beddings, that is we use the word embedding vector of either only the current word or along with its two neighbouring words. The latter case also requires zero padding if no neighbouring word is present, as is the case at the end of an utterance. We test different bottleneck sizes n, so that the final feature representation along with the acoustic features of size $x = 100$ has the dimensionality $100 + n$. The sizes are set such that the output of the lexical model is much smaller than that of the acoustic model, for example $n = 10, 20, 30$. By restricting the number of lexical features, we aim to ensure that it does not overpower the acoustic information.

5. Experiments

5.1. Within-corpus and cross-corpus training

The models are trained usind 10-fold cross-validation for 20 epochs with an adaptive learning rate (Adam [26]). After training, the best model according to a held out validation set is applied to the test set. All experiments are repeated 5 times and the results are averaged.

It is to be noted that the splitting of the corpora into training and test sets is carried out randomly, which means that the within-corpus experiments are neither evaluated independent of speaker nor of lexical content. The latter can be found especially in the BURNC and LeaP corpora, in which different speakers often read the same text. In the LeaP dataset, around half of the utterance files were recorded from speakers reading out the same story. Since the experiments focus on adding lexical information, we do expect the model to learn observed textual features. For this reason, strictly unseen data is tested in the cross-corpus experiments, using entirely different corpora.

For the within-corpus experiments, we split the datasets into 10 fixed cross-validation splits, and use 1000 held-out words from the respective training split as the development set. In the cross-corpus experiments, the same 10 test sets are used for comparability with the within-corpus versions. Training is carried out on an entire source corpus and the development set consists of 1000 held-out words from this dataset. In the *ALL* setting, the separate target training set for each target test split is added to the training data.

5.2. Results

5.2.1. Baseline

Table 3 shows the accuracy obtained in the within-corpus and cross-corpus experiments. As expected, we observe that the performance in the cross-corpus settings drops when compared to the within-corpus setting (BURNC 87.1%, BDC 77.9%, LeaP 80.4%). BURNC appears to be by far the easiest corpus for pitch accent detection, resulting in accuracy levels obtained while training on BDC or LeaP that are higher than those obtained on the respective within-corpus test sets. The *ALL* experiments show a slight rise in accuracy that does not quite reach the baseline within-corpus performance, except on LeaP (80.5%). The BDC datasets appears to be the most difficult to model using this method. Overall, the acoustic model can be said to generalize reasonably well across corpora and genres and the within-corpus detection accuracy can be compared to that reported in related work: The results on BURNC, outperform previous methods [3]. The performance on BDC is slightly higher than the results obtained using acoustic features and speaker-independent evaluation in [27]. The accuracy on the LeaP corpus is similar to previously reported numbers obtained at the syllable level [23], although these experiments are not directly comparable.

Table 3: *Results (accuracy in %) for within-corpus and cross-corpus pitch accent detection using the baseline acoustic model, with added unigram GloVe word embeddings and 10 bottleneck features, and using only the embeddings as features (shown in italics). The results are averaged across 10 cross-validation splits and 5 repetitions of each experiment.*

Train \ Test	BURNC	BDC	LeaP
BURNC			
acoustic	87.1	74.8	79.3
acoustic+embs	**87.5**	74.2	79.0
embs-only	*78.5*	*71.8*	*76.2*
BDC			
acoustic	83.0	77.9	76.6
acoustic+embs	81.3	**78.4**	76.2
embs-only	*75.2*	*76.3*	*74.6*
LeaP			
acoustic	82.8	73.2	80.4
acoustic+embs	81.8	72.7	**81.5**
embs-only	*67.2*	*68.0*	*81.2*
ALL			
acoustic	86.6	77.0	80.5
acoustic+embs	87.0	77.9	81.2
embs-only	*75.2*	*72.1*	*77.7*

5.2.2. Effect of adding word embeddings

The first experiments compare the effect of adding word embeddings to within-corpus and cross-corpus settings. We use unigram word embeddings and 10 bottleneck features because these are taken to be less corpus-specific than trigram features. Table 3 contains only the results using GloVe embeddings since the results using word2vec are similar. We observe an increase in accuracy on all corpora (marked in bold) in the corpus-dependent setting, which shows that the model can learn representations from word embeddings that are helpful for detecting pitch accents. In contrast, the results obtained on cross-corpus experiments lead to the conclusion that adding word embeddings to the baseline model harms its generalization performance. All numbers except those for the *ALL* experiments show a slight decrease in accuracy. This is due to the strong lexical features causing the model to overfit to the respective training

Table 4: *Results (accuracy in %) for within-corpus pitch accent detection with word embeddings using unigrams, trigrams and varying bottleneck sizes.*

Corpus	BURNC		BDC		LeaP	
Embeddings	glove	w2v	glove	w2v	glove	w2v
unigram						
$n = 10$	87.5	87.6	78.4	78.4	81.5	81.7
$n = 20$	87.5	87.5	78.4	78.5	81.6	82.0
trigram						
$n = 10$	87.7	87.6	78.5	78.3	81.5	81.7
$n = 30$	87.8	87.6	78.5	78.5	81.7	81.8

Table 5: *Precision, recall and F1 scores for the accent class using unigram word embeddings and 10 bottleneck features.*

Experiment	Precision	Recall	F1
BURNC			
baseline	87.8	87.1	87.4
w2v	87.0	89.5	88.2
GloVe	87.9	87.9	87.9
BDC			
baseline	74.1	78.6	76.2
w2v	73.3	81.9	77.3
GloVe	73.6	81.2	77.2
LEAP			
baseline	77.1	78.2	77.5
w2v	78.3	80.1	79.2
GloVe	77.7	80.7	79.2

corpus. Only when target corpus data held out from the test set is seen in training (*ALL*) do we observe better results. In fact, the performance of *ALL* on LeaP is almost as high as the within-corpus results using word embeddings (81.5%). The results on BURNC and BDC are similar to the respective baseline within-corpus performance. The embeddings-only results were added to give an impression of how these features would perform alone. In all cases, they are not as strong as the acoustic features alone, except on the LeaP corpus. This confirms that the lexical overlap in LeaP greatly facilitates the detection of pitch accents.

5.2.3. Effect of N-gram and bottleneck sizes

In Table 4 we show results when adding unigram and trigram embeddings to the baseline acoustic model. We also test different bottleneck sizes n. The word embeddings improve the accuracy over the baseline on all corpora across all tested settings. Overall, the results do not show a specific pattern, from which it can be concluded that neither the choice of N-gram and bottleneck size nor the embedding type is critical in our experiments. The corpus that appears to benefit the most from these lexical features is the LeaP corpus. Here we observe an increase in over 1 percentage point in all experiments. To provide more insight into these numbers, we also show the precision, recall and F1 scores for the accent class in Table 5 when using unigram embeddings and 10 bottleneck features. The word embeddings lead to a gain in F1, precision and recall in all cases, except on BDC, where the precision decreases slightly.

Table 6: *Stopword accuracy (%) using unigram embeddings and 10 bottleneck features.*

	BURNC	BDC	LeaP
baseline	98.1	50.0	75.4
GloVe	98.2	50.4	81.0
w2v	98.1	48.5	82.6

5.2.4. Performance on stopwords

In word2vec, stopwords are represented as OOV with a zero vector, wheras in GloVe, they are part of the vocabulary. Because of this difference we also compute the performance accuracy just on stopwords (Table 6). On the LeaP corpus, we observe better performance on these words with both types of embeddings. It appears that stopwords are more likely to be labelled correctly (which is usually not accented) when there is some information about the word identity available, regardless of how they are represented as vectors. An unexpected result is that only around half of the stopwords are predicted correctly in BDC. Further investigation is necessary to establish why the performance on stopwords in this corpus is poor. This issue may also be partly explained by the fact that this corpus has the highest rate of accented stopwords (see Table 2).

6. Discussion

The baseline model alone appears to be reasonably strong, despite the fact that the input features require very little preprocessing and linguistic knowledge. Further analysis and optimization steps may help to increase the performance on the BDC dataset. Within-corpus experiments show that simply adding word embeddings does provide information that the model can use to learn useful features that correlate with pitch accents, as the gain in accuracy in these cases demonstrates. We find, however, that the challenge lies in that such strong features tend to lead to overfitting. The fact that the more textual overlap is found in training and test data, the better the increase in performance, must be kept in mind when using word embeddings in such a setting. Possible future directions include the exploration of how the acoustic and lexical information interact during training, and what lexical information the model is learning to exploit: e.g. to what extent semantic or syntactic information is encoded in the bottleneck features.

7. Conclusion

In this paper, we investigate the effect of adding word embeddings as a simple method of exploiting lexical information in an extension of a CNN-based pitch accent detector. The results from within-corpus experiments show that the model can clearly benefit from word embeddings, but that generalizing to unseen datasets remains a challenging task. Cross-corpus experiments show that the baseline acoustic model alone yields good results. Future research can help ascertain what type of information the model is learning from word embeddings.

8. Acknowledgements

This work is funded by the Sonderforschungsbereich (collaborative research center) SFB-732 of the German National Science Foundation (DFG).

9. References

[1] C. Wightman and M. Ostendorf, "Automatic labeling of prosodic patterns," in *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, 1994, pp. 469–481.

[2] J. Hirschberg, "Pitch accent in context: Predicting intonational prominence from text," *Artificial Intelligence*, vol. 63, pp. 305–340, 1993.

[3] A. Rosenberg, "Automatic detection and classification of prosodic events," Ph.D. dissertation, Columbia University, 2009.

[4] A. Rosenberg, E. Cooper, R. Levitan, and J. Hirschberg, "Cross-language prominence detection," in *Speech Prosody*, 2012.

[5] A. Schweitzer and B. Möbius, "Experiments on automatic prosodic labeling," in *Proceedings of Interspeech*, 2009, pp. 2515–2518.

[6] S. Ananthakrishnan and S. S. Narayanan, "Automatic prosodic event detection using acoustic, lexical and syntactic evidence," in *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 1, 2008, pp. 216–228.

[7] X. Sun, "Pitch accent prediction using ensemble machine learning," in *Proceedings of ICSLP-2002*, 2002, pp. 16–20.

[8] K. Chen, M. Hasegawa-Johnson, and A. Cohen, "An automatic prosody labeling system using ann-based syntactic-prosodic model and gmm-based acoustic-prosodic model," in *Proceedings of ICASSP*, 2004, pp. 509–512.

[9] S. Pan and K. R. McKeown, "Word informativeness and automatic pitch accent modeling," in *In Proceedings of EMNLP/VLC*, 1999, pp. 148–157.

[10] J. Yuan, J. M. Brenier, and D. Jurafsky, "Pitch accent prediction: Effects of genre and speaker," in *Proceedings of Interspeech*, 2005, pp. 1409–1412.

[11] A. Margolis, M. Ostendorf, and K. Livescu, "Cross-genre training for automatic prosody classification," in *Speech Prosody*, 2010.

[12] A. Nenkova, J. Brenier, A. Kothari, S. Calhoun, L. Whitton, D. Beaver, and D. Jurafsky, "To memorize or to predict: Prominence labeling in conversational speech," in *in Proceedings of NAACL-HLT*, 2007, pp. 9–16.

[13] S. Stehwien and N. T. Vu, "Prosodic event recognition using convolutional neural networks with context information," in *Proceedings of Interspeech*, 2017.

[14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the Workshop at ICLR*, 2013.

[16] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *HLT-NAACL*, 2013.

[17] P. Wang, Y. Qian, F. Soong, L. He, and H. Zhao, "Word embedding for recurrent neural network based tts synthesis," in *ICASSP-2015*, 2015, pp. 4879–4883.

[18] A. Rendel, R. Fernandez, R. Hoory, and B. Ramabhadran, "Using continuous lexical embeddings to improve symbolic-prosody prediction in a text-to-speech front-end," in *ICASSP 2016*, 2016, pp. 5655–5659.

[19] K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, and C. Wightman, "ToBI: A standard for labelling English prosody," in *Proceedings of ICSLP*, 1992, pp. 867–870.

[20] M. Ostendorf, P. Price, and S. Shattuck-Hufnagel, "The Boston University Radio News Corpus," Boston University, Technical Report ECS-95-001, 1995.

[21] J. Hirschberg and C. H. Nakatani, "A prosodic analysis of discourse segments in direction-giving monologues," in *IN 34TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 1996, pp. 286–293.

[22] J. Torsten Milde and U. Gut, "A prosodic corpus of non-native speech," in *Speech Prosody*, 2002.

[23] G. Anne Levow, "Investigating pitch accent recognition in non-native speech," in *Proceedings of ACL-IJCNLP*, 2009, pp. 269–272.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] F. Eyben, F. Weninger, F. Groß, and B. Schuller, "Recent developments in opensmile, the Munich open-source multimedia feature extractor," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 835–838.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.

[27] V. Kumar, R. Sridhar, S. Bangalore, and S. S. Narayanan, "Exploiting acoustic and syntactic features for automatic prosody labeling in a maximum entropy framework," in *IEEE Transactions on Audio, Speech & Language Processing*, vol. 16, no. 4, 2008, pp. 797–811.