Interspeech 2018
2-6 September 2018, Hyderabad

Entity-Aware Language Model as an Unsupervised Reranker

Mohammad Sadegh Rasooli[1], *Sarangarajan Parthasarathy*[2]

[1]Department of Computer Science, Columbia University, New York, NY, 10027
[2]Artificial Intelligence and Research, Microsoft Corporation, USA

rasooli@cs.columbia.edu, sarangp@microsoft.com

Abstract

In language modeling, it is difficult to incorporate entity relationships from a knowledge-base. One solution is to use a reranker trained with global features, in which global features are derived from n-best lists. However, training such a reranker requires manually annotated n-best lists, which is expensive to obtain. We propose a method based on the contrastive estimation method that alleviates the need for such data. Experiments in the music domain demonstrate that global features, as well as features extracted from an external knowledge-base, can be incorporated into our reranker. Our final model, a simple ensemble of a language model and reranker, achieves a 0.44% absolute word error rate improvement over an LSTM language model on the blind test data.

1. Introduction

Voice assistant systems rely heavily on complex language models. These language models are used as a second-pass reranking step for hypotheses generated by a first-pass speech recognizer [1, 2, 3, 4]. While a significant fraction of queries in voice assistant systems relate to entities in the real world (such as names of places, products, arts, people, etc.), language models do not explicitly model them. The problem of recognizing entities becomes more critical when some entities are not adequately represented in the training data. For example, Figure 1 shows two possible outputs from a first-pass speech recognition system. The correct output is about playing a song by a specific singer. The singer and the song have not been seen in the training data together, but by incorporating the cross-entity relationship from a relevant knowledge-base, we can capture the correct output of the speech recognition system.

Unlike previous work [5, 6] that have considered modeling entities directly in a language model, this paper proposes a dynamic reranking approach without the need for any transcribed training data. Our model makes use of raw text and a knowledge-base consisting of entities in the world. In order to train a reranker, we create artificial n-best lists for each training sentence. This enables us to train a reranking model on the artificial n-best list. We use the contrastive estimation method [7][1] to maximize the likelihood of each sentence in the training data in contrast to artificial sentences in the n-best list. One main strength of our method is that we are not bound to local word-based features anymore, thereby facilitating the possibility of embedding global features such as phrase-based interactions between entities (such as the "played-by" relationship in Figure 1).

This paper considers using language modeling as a global reranking approach in which the reranker makes use of many features including the bidirectional LSTM-based representations of sentences, n-gram language model probability, cross-entity relationships and frequency of entities in the knowledge-base. In other words, we maximize the probability of the whole sentence given its artificial negative examples. By employing this approach, we are able to improve the word error rate of our in-house speech recognition system by $0.44\,\%$ absolute difference. This reduction is in particular very interesting to us, because we only target one aspect of the language modeling problem.

[1]This method is different from the noise contrastive estimation method [8].

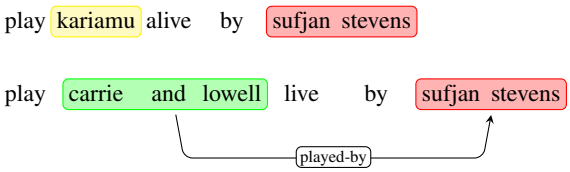

Figure 1: *An example of two possible outputs by the speech recognition system. The LSTM-based language model predicts the first option while our model prefers the correct output.*

In summary, the main contributions of this paper are as following:

- Designing a reranker, based on global features, to incorporate entities in the language model.
- Proposing an effective approach for generating artificial n-best lists for training a reranker. By applying this idea, we do not need to have transcribed training data.
- Introducing features from a knowledge-base and showing their effectiveness in the performance of our speech recognition system.

The remainder of this paper is as following: §2 briefly describes the background on language modeling. §3 describes our approach and §4 shows the experimental results. Finally, we conclude in §5.

2. Background

A recurrent language model uses a recurrent function RNN and derives the intermediate representation $h_i = RNN(h_{i-1};\theta) \in \mathbb{R}^H$ for every word x_i in sentence X. θ is a set of parameters in the recurrent model.

$$p(X;\theta) = \prod_{i=1}^{n} p(x_i|h_{i-1};\theta) = \prod_{i=1}^{n} \sigma_{x_i}(Wh_{i-1} + b) \quad (1)$$

In Eq. 1, σ is the softmax function. $W \in \mathbb{R}^{V \times H}$ and $b \in \mathbb{R}^V$ are the output layer and bias term, and V is the vocabulary size.

Recent studies [3, 9, 10] have shown that recurrent neural network (RNN) language models perform significantly better than n-gram language models. Long short-term memories (LSTM) [11] are the most popular RNN functions used in the

10.21437/Interspeech.2018-62

jim dandy's near me noblesville indiana

gym dandies near me noblesville indiana

jim dandys near me noblesville indiana

jim dandes near me noblesville indiana

jim dandies near me noblesville indiana

Figure 2: *An example of a correct sentence on top and its implicit negative examples. The words highlighted in red are randomly replaced based on their phonetic similarity.*

language modeling problem. Besides all the benefits in using recurrent models, they can only compute sentence probabilities in word space; this is an important problem especially for multi-word entities.

Representing entities in language models has been considered in previous work. For example, entities are modeled by [5, 12] as additional information in an ngram language model. Ahn et al. [6] uses a fact-based model to incorporate information available in a knowledge base. In contrast to their word-based model, our model can capture global information beyond words. Ji et al. [13] uses a dynamic model to incorporate multiple entities while processing the data. Their model achieves a slight improvement in perplexity, but it is not clear if their model can improve the error rate on big datasets. The recent work by Biadsy et al. [4] shows the effectiveness of log-linear models with global features using transcribed data. We instead use the contrastive estimation method [7] to maximize the probability of a correct sentence given its implicit negative examples. This enables us to use global features in our model, as well as word-based features, without using any transcribed data.

3. Approach

This section describes our approach based on the contrastive estimation method [7]. We list the features and describe the neural network architecture that we use in our model.

3.1. Data Assumption

We assume availability of the following sources for training the entity-aware language model:

Raw text: A large amount of raw text $\mathcal{X} = \{x^{(1)}, \ldots, x^{(m)}\}$ where $x^{(j)}$ is the jth sentence in the dataset. Each sentence $x^{(j)}$ consists of l_j words: $\{x_1^{(j)}, \cdots, x_{l_j}^{(j)}\}$. In this paper, we train our model on music queries. These queries are usually about asking a voice system to play or download some particular music.

Knowledge-base: A database $\mathcal{K} = \{k^1, \cdots, k^n\}$ such that each entry k^j in the knowledge-base consists of m fields $k^j = \{k_1^j, \cdots, k_m^j\}$. In this paper, we use a knowledge-base that consists of music information. We use three fields: artist name, song title, and the frequency of usage of the song in our in-house application.

3.2. Contrastive Estimation

Contrastive estimation [7] requires creation of *implicit* negative examples for each training sample. This is done by injecting artificial noise to the correct example. Therefore, for every sentence $x^{(j)} \in \mathcal{X}$ in the training data, we approximate its probability with respect to the negative examples $\mathcal{N}(x^{(j)})$ and model parameter θ.

$$P(x^{(j)}; \theta) \approx P(x^{(j)} | \mathcal{N}(x^{(j)}); \theta) = \frac{e^{u(x^{(j)};\theta)}}{\sum_{x' \in \{\mathcal{N}(x^{(j)}) \cup x^{(j)}\}} e^{u(x';\theta)}} \quad (2)$$

where $u(x^{(j)}; \theta)$ is the scoring function of the sentence $x^{(j)}$ given the model parameter θ. The objective function for the training data $\mathcal{X}$ is the following:

$$L(\mathcal{X}; \theta) = -\sum_{j=1}^{m} \log P(x^{(j)}; \theta) + \lambda \left\|\theta\right\|_2 \quad (3)$$

where λ is a constant coefficient for L2 regularization.

3.2.1. Creating Negative Examples

The definition of the negative example function depends on the task. Since we target outputs from a voice system, our observation shows that most of the real errors come from a confusion by the model between two phonetically similar words or phrases. We use a simple phonetic similarity function to randomly pick words in a training sentence and replace them with one of their phonetically similar words. We rerank the negative samples based on their n-gram language model probability and pick the five highest scoring ones. This is mainly because the real n-best lists usually consist of relatively fluent sentences while so many of the negative samples are not fluent sentences. By applying this reranking step, we avoid totally irrelevant negative samples. Figure 2 shows a real example of the negative examples created by our method.

3.2.2. Comparison to Noise Contrastive Estimation (NCE)

Noise contrastive estimation [8] is a popular method in language modeling. In this method, the probability of a word is maximized given its negative samples. The negative samples come from a probability distribution aside from the current model. Then the method is defined as a binary classification problem in which the label for the correct word is one and for the negative examples is zero. NCE has interesting properties such as being self-normalized. One challenge in NCE is that one should be able to define a well-formed probability distribution for negative examples. This is very straightforward for word-level language modeling; for example, we can define the noise distribution as the categorical distribution derived from the word counts in the training data. In our case, we are interested in changing more than one word in a sentence to create negative examples. That makes it hard for us to define a well-formed probability distribution for negative examples. On the other hand, although contrastive estimation is not as principled as NCE, we do not have a strict limitation on defining the negative examples using contrastive estimation.

3.3. Features

For a sentence x with l words, the following features are used:

- **Recurrent representation:** We use a bidirectional LSTM (BiLSTM) β to compute the sentence-level representation for each sentence. In other words, we have two independent LSTMs, one for the forward pass that sweeps a sentence from left to right and the other for the

backward pass that does a reverse sweep of the sentence. The forward and backward LSTMs give the following outputs:

$$\begin{aligned}[E[x_1], \ldots, E[x_l]] &\xrightarrow{f-LSTM} [h_1^f, \cdots, h_l^f] \\ [E[x_l], \cdots, E[x_1]] &\xrightarrow{b-LSTM} [h_l^b, \cdots, h_1^b]\end{aligned} \tag{4}$$

where $E[x_i]$ is the word embedding vector for word x_i. We use the concatenation of the final representations as the recurrent representation of the sentence. The LSTM parameters are updated during training with backpropagation:

$$\phi^{rnn}(x; \beta, E) = [h_l^f; h_1^b] \tag{5}$$

- **N-gram LM probability:** This score is fixed during training:

$$\phi^{ngram}(x; \theta_{ngram}) = -\log P(x; \theta_{ngram}) \tag{6}$$

The ngram language model feature is obtained with 10-way jack-knifing in order to avoid overfitting to the training data.

- **Phrase pair co-occurrence:** For every pair of non-overlapping phrases in sentence x, we count the number of entries in the knowledge-base that has the first sub-phrase as an *artist* and the second as a *song name* (or vice versa). Finally, we quantize this value into a bin of size 10 (based on the maximum possible co-occurrence count in the knowledge-base) and embed that to a embedding dictionary $\gamma \in \mathbb{R}^{10 \times d_m}$. Thus the cross-entity phrase co-occurrence feature $\phi^{co}(x; \gamma, \mathcal{K})$ is a d_m-dimensional vector.

- **Subsentence knowledge-base frequency:** We compute the sum of *frequency field* of each phrase in a sentence x for the artist and song fields.

$$\begin{aligned}f^1(x; \mathcal{K}) &= \log \sum_{i=1}^{l} \sum_{j=i}^{l} freq_{artist}^{\mathcal{K}} x[i:j] \\ f^2(x; \mathcal{K}) &= \log \sum_{i=1}^{l} \sum_{j=i}^{l} freq_{song}^{\mathcal{K}} x[i:j]\end{aligned} \tag{7}$$

We quantize these values into the integer range $[0, 100)$ and represent them as embedding vectors $\mu, \nu \in \mathbb{R}^{100 \times d_{c_f}}$:

$$\begin{aligned}c_f^1(x; \mathcal{K}) &= \left\lfloor min(50 \times f^{artist}(x), 99) \right\rfloor \\ c_f^2(x; \mathcal{K}) &= \lfloor min(50 \times f^{song}(x), 99) \rfloor\end{aligned} \tag{8}$$

The final features are $\phi^1(x; \mu, \mathcal{K})$ and $\phi^2(x; \nu, \mathcal{K})$ as two d_f dimensional vectors.

- **Cross-entity and Intra-entity word-based mutual information:** We observed that many phrasal entities have different forms (such as using abbreviations for first names). Therefore, it can be useful to incorporate word-level features for words in entities. This is done by enumerating the mutual information between words across the artist and song fields. We use the average of normalized pointwise mutual information [14] between words.

$$in(x; \mathcal{K}) = \frac{\sum_{i=1}^{l-1} \sum_{j=i+1}^{l} -\log \frac{p^{\mathcal{K}}(x_i, x_j)}{p^{\mathcal{K}}(x_i) p^{\mathcal{K}}(x_j)} / \log p^{\mathcal{K}}(x_i, x_j)}{l \cdot (l-1)/2} \tag{9}$$

where the probabilities $p^{\mathcal{K}}$ are calculated based on the frequency information in the knowledge-base. We scale the value $in(x, k)$ to be in $[0, 1]$ and then quantize it into 100 bins:

$$f_{in}(x; \mathcal{K}) = \left\lfloor 100 \cdot \frac{in(x) + 1}{2} \right\rfloor \tag{10}$$

Finally, we embed this to an embedding parameter $\eta \in \mathbb{R}^{100 \times d_{xp}}$. $\phi^{xp}(x; \eta, \mathcal{K}) \in \mathbb{R}^{d_{xp}}$ is the embedding feature for representing the cross-entity word-based mutual information. Similarly, we use an embedding dictionary $\kappa \in \mathbb{R}^{100 \times d_{ip}}$ and $\phi^{ip}(x; \kappa, \mathcal{K}) \in \mathbb{R}^{d_{ip}}$ for the feature representing intra-entity word-based mutual information.

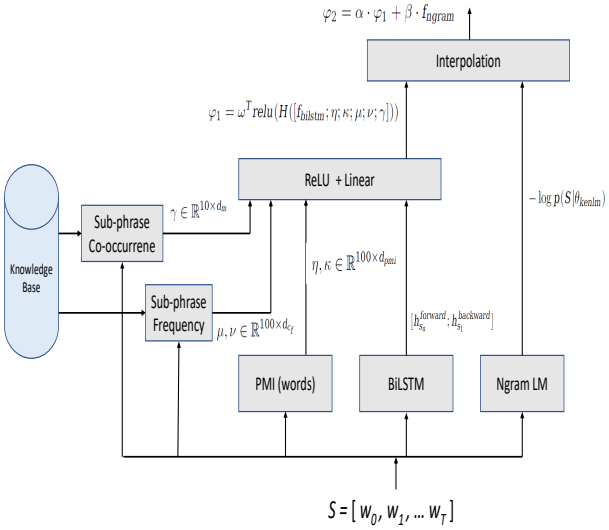

Figure 3: *A graphical depiction of the scoring function for the reranker model. Here, we assume that the knowledge base and the ngram model θ_{ngram} is already given to us, and the model will learn the embedding and dense parameters using the negative examples during training. The parameters $E, \gamma, \nu, \mu, \eta, \kappa$ are embedding parameters, β is the set of BiLSTM parameters; and H and ω are dense layers. The scores from the ngram model is interpolated with the final score from the deep model. In training, there is a softmax layer on top that defines the contrastive estimation loss with respect to the correct output in contrast to the negative examples in the n-best list.*

3.4. Network Architecture

All the features, except the ngram probability, are concatenated as $\phi(x; \theta, \mathcal{K})$ and fed to a hidden layer with a rectified linear unit (RELU) [15] activation. The output from the hidden layer is multiplied by a vector ω:

$$s(x;\theta,\mathcal{K}) = \omega^T relu(H^T \phi(x;\theta,\mathcal{K})) \quad (11)$$

Finally, we use a linear interpolation between $h(x,\theta)$ and the n-gram feature $\phi^{ngram}(x;\theta)$. All parameters in this model, except the ngram probabilities, are tuned during backpropagation:

$$u(x;\theta,\mathcal{K},\theta_{ngram}) = \alpha_1 \cdot s(x;\theta,\mathcal{K}) + \alpha_2 \cdot \phi^{ngram}(x;\theta_{ngram}) \quad (12)$$

where during training, we use the parameter u in the above equation (Eq. 12) to calculate the approximate probability of x with respect to negative examples as in Eq. 2.

Figure 3 shows a graphical depiction of the network structure.

4. Experiments

4.1. Data and Setting

Our pipeline uses a domain classifier to classify queries in a query stream. We select queries tagged as belonging to the music domain. We mapped words with frequency less than one in the training data to the *unknown* symbol. Our heldout data is a small set of transcribed queries (3941 sentences) from a mobile phone application and our blind test set is from a distant microphone with relatively poor quality speech signals (4970 sentences). Both the heldout and test data have five hypotheses per sentence. The knowledge-base consists of 13 million entries. For each entry, we select the title of the song, artist name, and the frequency of requests. The training data has 5.2 million sentences consisting of 24 million words. The vocabulary size after converting infrequent words to the unknown symbol is 129935.

4.2. Negative examples

We first train the KenLM ngram language model [16] with 10-way jackknifing on the training data. We sample 30 random negative examples for each sentence in the training data and use the probabilities from the language model to get the 5 best examples. Our observation shows that the quality of some of the negative examples are still not promising. Therefore, we just kept the top one million training examples where the average ngram probability of their negative examples are highest. We train the same ngram model on the training data to calculate probabilities on the heldout and test data.

4.3. LSTM language model and reranker

Since our reranker just uses a portion of the training data, we might lose some performance from having a smaller training data. Therefore, the final score of each sentence in decoding is summed with the score from an LSTM model trained on the whole training data. To train the standard language model with LSTM, we use noise contrastive estimation [8] with 100 fixed samples per each minibatch of 64 sentences to train our model. The bias term is initialized as $-\log N$ where N is the vocabulary size. We use L2-regularization with coefficient 10^{-6}. Stochastic gradient descent with momentum [17] and learning rate of 1.0 with momentum 0.9 and decay 0.5 is used to train the model parameters. We apply dropout [18] with probability 0.2 in training. Word embedding vectors are initialized randomly with dimension 200 and the LSTM dimension is 1000.

We use a similar LSTM model in both directions inside the reranker but with a smaller dimension for the LSTM representation (500). Early stopping is applied based on word error rate improvement on the heldout data. We use dimension of 50 for d_f, d_{xp} and d_{ip} and 10 for d_m. We use the Dynet library [19] for implementing all of our models.

Model	heldout	Test
First-pass (no reranker)	11.52	12.52
Ngram LM	11.17	11.56
LSTM LM	10.16	11.56
Reranker	10.03	11.35
Reranker + LSTM	**9.82**	**11.12**
Oracle	6.55	7.32

Table 1: *Experimental results based on word error rate (WER) on the heldout and test data. First-pass shows the model accuracy by just applying the first-pass language model from the speech recognition system. The other rows show the effect of interpolating it with different types of language models or reranker. Oracle shows the WER when we pick the sentence in the n-best list with the lowest WER with respect to the gold-standard output.*

4.4. Results

Table 1 shows the experimental results on the heldout and test data. As shown in the table, the reranker, with only one-fifth of the real training data, perform better than both the LSTM and ngram language models. When the reranker is merged with the LSTM language model, the performance improves further. We believe that is due to the bigger size of the training data for the LSTM model. The final ensemble result achieves an absolute 0.44% improvement compared to the LSTM language model on the test data. Another observation is that LSTM LM is more effective than ngram LM for reranking on the heldout set but not on the test set (rows 2 and 3 in Table 1). A possible explanation of this result is the difference in quality of the n-best hypotheses between heldout and test sets. Recall that the speech for heldout set is collected from mobile phones whereas the test set data is collected from a distant device.

5. Conclusion

In this paper, we have shown an effective method to encode real-world entities into a language model. We designed a simple but effective approach to create artificial n-best lists, thus obviating the need for annotated data. Our experiments show an improvement on the heldout and test datasets. One interesting direction to pursue is to incorporate a small amount of transcribed data and use our approach on a combined set of transcribed dataset and artificial training data. There are certain challenges facing this approach when dealing with a mixture of real n-best lists and artificial n-best lists such as deciding about the proportion of real n-best lists compared to the artificial ones. Future work should consider studying this problem.

6. Acknowledgement

We thank the anonymous reviewers for their valuable feedback. This research was conducted while the first author was an intern in the language modeling group at Microsoft in Sunnyvale, California. We would like to thank the researchers in the group for helpful discussions and assistance on different aspects of the problem.

7. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[3] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *arXiv preprint arXiv:1707.05589*, 2017.

[4] F. Biadsy, M. Ghodsi, and D. Caseiro, "Effectively building tera scale maxent language models incorporating non-linguistic signals," *Interpspeech 2017*, 2017.

[5] M. Levit, A. Stolcke, S. Chang, and S. Parthasarathy, "Token-level interpolation for class-based language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5426–5430.

[6] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, "A neural knowledge language model," *arXiv preprint arXiv:1608.00318*, 2016.

[7] N. A. Smith and J. Eisner, "Contrastive estimation: Training log-linear models on unlabeled data," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 354–362.

[8] M. Gutmann and A. Hyvrinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 297–304. [Online]. Available: http://proceedings.mlr.press/v9/gutmann10a.html

[9] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[10] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model." *SLT*, vol. 12, pp. 234–239, 2012.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] M. Levit, A. Stolcke, R. Subba, S. Parthasarathy, S. Chang, S. Xie, T. Anastasakos, and B. Dumoulin, "Personalization of word-phrase-entity language models," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[13] Y. Ji, C. Tan, S. Martschat, Y. Choi, and N. A. Smith, "Dynamic entity representations in neural language models," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 1831–1840. [Online]. Available: http://www.aclweb.org/anthology/D17-1195

[14] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction," *Proceedings of GSCL*, pp. 31–40, 2009.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[16] K. Heafield, "KenLM: faster and smaller language model queries," in *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland, United Kingdom, July 2011, pp. 187–197. [Online]. Available: https://kheafield.com/papers/avenue/kenlm.pdf

[17] G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_32

[18] N. Srivastava, G. E. 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.

[19] G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn *et al.*, "Dynet: The dynamic neural network toolkit," *arXiv preprint arXiv:1701.03980*, 2017.