Comparing Character-level Neural Language Models Using a Lexical Decision Task

Gaël Le Godais[1,3] **Tal Linzen**[1,2] **Emmanuel Dupoux**[1]
[1]LSCP, CNRS, EHESS and ENS, PSL Research University [2]IJN [3]ENSIMAG
gael.le-godais@orange.fr, {tal.linzen, emmanuel.dupoux}@gmail.com

Abstract

What is the information captured by neural network models of language? We address this question in the case of character-level recurrent neural language models. These models do not have explicit word representations; do they acquire implicit ones? We assess the lexical capacity of a network using the lexical decision task common in psycholinguistics: the system is required to decide whether or not a string of characters forms a word. We explore how accuracy on this task is affected by the architecture of the network, focusing on cell type (LSTM vs. SRN), depth and width. We also compare these architectural properties to a simple count of the parameters of the network. The overall number of parameters in the network turns out to be the most important predictor of accuracy; in particular, there is little evidence that deeper networks are beneficial for this task.

1 Introduction

Neural networks have rapidly become ubiquitous in natural language processing systems, but our ability to understand those networks has not kept pace: we typically have little understanding of a typical neural network beyond its accuracy on the task it was trained to do. One potential way to gain insight into the ability of a trained model is to evaluate it on an interpretable auxiliary task that is distinct from the task that the network was trained on: a network that performs a particular auxiliary task successfully is likely to have internal representations that encode the information relevant for that task (Adi et al., 2017; Mikolov et al., 2013). Linguistics and psycholinguistics offer a rich repertoire of tasks that have been used for decades to study the components of the human mind; it is natural to use these tasks to understand the abilities of artificial neural networks (Dunbar et al., 2015; Linzen et al., 2016).

The present work takes up character-level neural network language models. Such models have been surprisingly competitive in applications, even though they do not explicitly represent words (Chung et al., 2016; Kim et al., 2016). Our goal is to shed light on the ability of character-level models to implicitly learn a lexicon. We use a task designed to investigate humans lexical processes. This task is based on a simple question: how well can the subject distinguish real words from character strings that do not belong to the language (nonwords)? Since character-level language models define a probability distribution over all character strings, we can perform this task in a particularly straightforward way: given a word and a nonword that are matched on low-level properties such as length and character bigram frequency, we expect the probability of the word to be higher than the probability of the nonword.

We systematically explore how the performance of the network on this task is affected by three architectural parameters. First, we vary the depth of the network (number of layers); second, we vary the number of units in each layer; and finally, we compare simple recurrent networks (SRN) to networks with long short-term memory cells (LSTM). We find that the main factor that determines the lexical capacity of the network is the total number of parameters rather than any one of these architectural properties.

2 Lexical decision

The lexical decision task is widely used in cognitive psychology to probe human lexical representations (Meyer and Schvaneveldt, 1971; Balota et al., 2006). In the standard version of the task, which we refer to as yes/no lexical decision, the subject is presented with a string of characters—e.g., *horse* in one trial or *porse* in another—and is requested to indicate whether or not the string makes up a word. A large array of properties of the word (or nonword) have been found to influence human performance on the task, measured in accuracy and reaction time; most famously, humans recognize frequent words more quickly and accurately than infrequent ones.

Our goal is to administer the lexical decision task to a character-level language model. Such a language model should assign a higher probability to words than to nonwords. At first blush, it appears straightforward to perform the task by fixing a probability threshold and classifying all of the strings whose probability falls above this threshold as words and all of the strings that fall below it as nonwords. In preliminary experiments, however, we found it difficult to define such a threshold. At a minimum, the probability assigned by the model to strings strongly depends on their length, so normalization for length is essential (see Lau et al. (2016) for discussion); even after normalization, however, it remained challenging to set a threshold distinguishing words from nonwords.

Instead of the standard yes/no lexical decision task, then, we use a forced choice variant of the task (Baddeley et al., 1993). In this version, two strings are simultaneously presented, one of which is always a word and the other always a nonword; subjects are instructed to select the item that they believe is a word. The advantage of this setup is that we can match each word with a nonword that is maximally similar to it in length or any other properties that may be relevant, thus avoiding complicated probability normalization schemes.

3 Models

We tested two types of recurrent units: the classic Elman (1990) architecture, which we refer to as simple recurrent network (SRN), and Long Short-Term Memory units, or LSTM (Hochreiter and Schmidhuber, 1997). Since each LSTM unit contains several gates and a memory cell, it has approximately four times as many connections as an SRN unit, and therefore four times as many parameters.

The first layer of each network is a character embedding. This layer is followed by one or more recurrent layers with a $\tanh$ nonlinearity, each followed by a batch normalization layer (Ioffe and Szegedy, 2015). A pair of 'view' layers then reshape the tensor with a linear transformation between them, yielding predicted scores for each element of the vocabulary. Finally, the output is produced by a softmax layer that gives a probability distribution over the next character.

How many parameters does each network have? Let n be its number of recurrent layers, V the size of the vocabulary (all possible characters), D the size of the character embedding, and H the number of units per layer. Table 1 shows the number of parameters in each layer:

Layer	Parameters
Character embedding layer	VD
First SRN layer	$H(D+H+1)$
First LSTM layer	$4H(D+H+1)$
Additional SRN layer	$H(2H+1)$
Additional LSTM layer	$4H(2H+1)$
Batch normalization layers	H
First 'view'	H
Linear transformation	HV
Second 'view'	V

Table 1: Number of parameters in each of the components of the model.

In addition to the RNNs, we test two simple baselines: a bigram and a unigram model of the training set. The goal of these baselines is to evaluate the nonwords: if a unigram model can reliably distinguish nonwords from words, the nonwords are not sufficiently challenging; this could happen, for example, if the nonwords tend to have rare characters such as Q or Z.

4 Methods

Corpus: We trained our language models on a subset of the movie book project corpus (Zhu et al., 2015); the subset contained approximately 50M characters (10M words). The corpus was lowercased by its creators. We split the corpus into training, validation and test sets (80%, 10% and 10% of the data, respectively); this test set was used only to calculate perplexity (see below). The vocabulary we used to test our network in the lexical decision task only included words that oc-

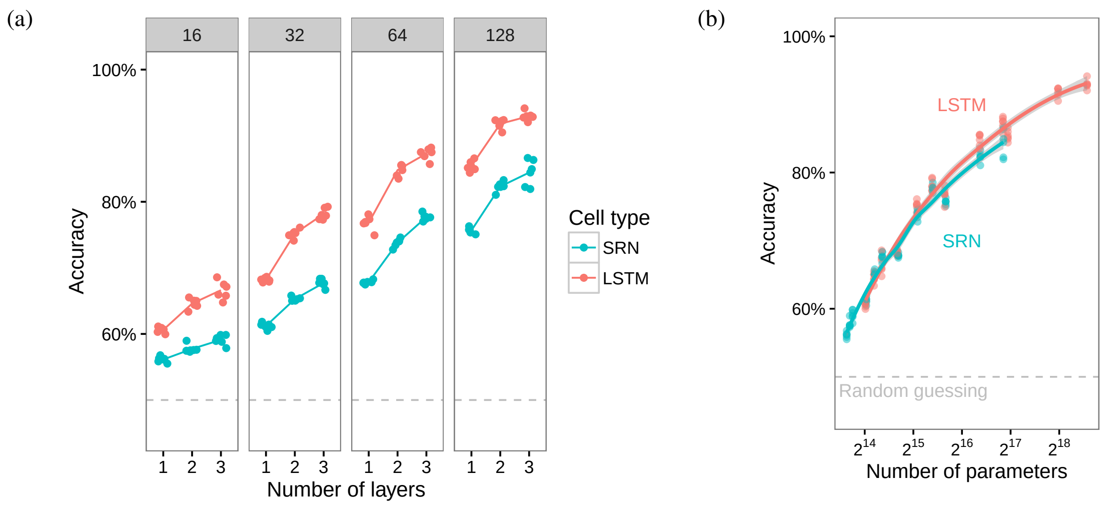

Figure 1: Accuracy as a function of the complexity of the network. The dashed line represents chance accuracy (50%). Each dot represents a single run.) (a) Detailed results by architecture, number of units per layer (16, 32, 64 or 128) and number of layers. (b) Relationship between accuracy and total number of parameters (on a logarithmic scale).

curred in the training set.[1]

Nonword generation: We generated nonwords using a slightly modified version of Wuggy (Keuleers and Brysbaert, 2010); we refer the reader to the original paper and our published code for further details.

The algorithm takes a list of words as its input and outputs a matching nonword for each word of the list. Matching is performed using a phonotactic grammar of the lexicon. This phonotactic grammar is based on a segmentation of the words into syllables and subsyllabic elements (onset, nucleus and coda). A syllabification dictionary splits the words into a sequence of syllables. Each syllable is then segmented into subsyllabic elements using a grammar of legal subsyllabic sequences. Each subsyllabic element is represented by a tuple that records its letters, position in the word, total number of subsyllabic elements in the word and the subsyllabic element that follows it. The first three elements of the tuples form a "link". The frequency of a link is computed from the lexicon, along with its possible next subsyllabic elements. This makes up a "bigram chain" that describes the phonotactics of the lexicon. For a given word, a nonword is generated by the bigram chain with parameters as similar as possible as the input word. Such parameters defined by the bigram chain can be, but are not limited to, the total length of the word and the transition probabilities between its subsyllabic elements.

Task: The RNN defines a probability distribution over character strings. We performed the forced choice task by calculating the probability of the word and the probability of the nonword, and selecting the string that had a higher probability; trials in which the probability of nonword was higher were considered to be errors. To ensure that we were computing the probability of a word rather than a prefix or suffix (e.g., *cat* as a prefix of *category*), we added spaces on either side of the word; e.g., we computed the probability of ' cat ' rather than 'cat'. We transformed the training corpus accordingly, to ensure that all words encountered during training contribute to the lexical decision, including words preceded or followed by a punctuation mark or a sentence boundary.

Experiments: We trained networks with all combinations of unit type (LSTM or SRN), width (16, 32, 64 or 128 hidden units per layer) and depth (one, two or three hidden layers). To estimate the impact of random initialization on the results, we trained six networks with each combination of parameters.[2]

We used a slightly modified version of Justin

[1]A network may be able to correctly perform a lexical decision on words to which it has not been exposed if those words follow the word formation rules of the language (e.g., *Frenchify*); we are exploring this issue in ongoing work.

[2]Our code can be found at `https://github.com/bootphon/char_rnn_lexical_decision`.

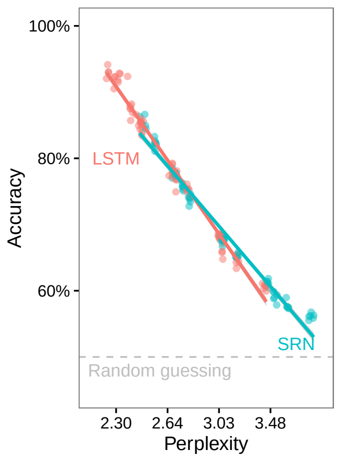

Figure 2: The relationship between character-level perplexity and lexical decision accuracy. Each point represent a single fitted model.

Johnson's Torch implementation of character-level RNNs.[3] To prevent overfitting, the networks were trained using early stopping based on validation set loss. They were optimized using Adam (Kingma and Ba, 2015) with a learning rate of $2e^{-3}$. The number of distinct characters was 95, and the dimension of the character embeddings was 64. During training, the networks operated over minibatches of size 50 and sequences of length 50.

5 Results

The accuracy of the unigram and bigram baselines was 49.6% and 52.1% respectively, very close to chance accuracy (50%). This suggests that the nonwords we generated were sufficiently difficult to distinguish from the words. The results of the RNNs we trained are shown in Figure 1a. All of the three architectural parameters affected performance in the task: networks with LSTM cells performed better than SRNs with the same number of units and layers. Increasing the number of units per layer was beneficial across the board. Additional layers improved performance as well, though the addition of the third layer was often less beneficial than the addition of the second one. Given a fixed budget of units, it was more useful to deploy them in a wide and shallow network than a narrow and deep network (e.g., an SRN with 32 hidden units in one layer outperformed an SRN with 16 hidden units in two layers).

How much of the advantage of LSTMs is due to the fact that they have more parameters per unit? Figure 1b plots the accuracy of the same networks, this time against the log-transformed number of parameters. While there remains a slight advantage for LSTMs over SRNs, especially as the number of parameters increases, it is evident that the number of parameters is an excellent predictor of the performance of the network. Of course, since the dependencies that the network needs to model to perform the lexical decision task are relatively short, this task may not bring out the competitive advantage of LSTMs, which are argued to excel in tasks that require long dependencies.

We plot the relationship between the perplexity of the language model and its accuracy in the lexical decision task in Figure 2. This relationship is not entirely surprising, given that low perplexity indicates that the model assigns high likelihood to the character sequences that occurred in the test set, which are of course much more likely to be words than nonwords. The two measures are far from being identical, however. Perplexity incorporates the model's ability to predict dependencies across words; this is not the case for lexical decision, where performance may in fact be hindered by irrelevant contextual information, as it is for humans (McDonald and Shillcock, 2001). Perplexity also weights accurate prediction of frequent words much more highly than infrequent words. Given these differences, the measures could potentially diverge in subsets of the lexicon.

6 Discussion

The lexical capacity measure that we have proposed assigns the same weight to rare and frequent words. As such, it may provide an alternative evaluation metric for character-based language models, supplementing the more standard measure of perplexity, which is biased in favor of frequent words and conflates lexical knowledge with longer dependencies across words.

One advantage of the evaluation metric we have proposed is that it is in principle possible to compare it to human performance. This contrasts with perplexity, which does not map onto any task that can be given to humans, especially when the model is at the character level. For example, our preliminary analyses showed that the model makes more errors on low-frequency than high-frequency words, a pattern that is qualitatively similar to hu-

[3]`https://github.com/jcjohnson/torch-rnn`

mans (Ratcliff et al., 2004).

Some challenges remain, however, before a quantitative comparison before humans and neural network language models can be performed. Existing large-scale human behavioral datasets are based on a speeded yes/no version of the task, in which participants are instructed to make a lexical decision on a single string of characters as quickly as possible (Balota et al., 2007), whereas our evaluation is based on the forced choice task and does not incorporate time pressure. A behavioral dataset with the paradigm we have used should be easy to collect using crowdsourcing. Alternatively, direct comparison to existing human datasets could be made possible by developing reliable ways to map language model probabilities onto timed yes/no lexical decisions; our initial experiments suggest that some nontrivial challenges would need to be overcome before this direction can be pursued.

Our work is related to early work that aimed to measure the phonotactic knowledge of recurrent networks (Stoianov et al., 1998; Stoianov and Nerbonne, 2000). This idea was developed by Testolin et al. (2016), who use the lexical decision task to measure the orthographic knowledge of various neural networks and n-gram models. The Naive Discriminative Learner (Baayen et al., 2011), which can be seen as a simple non-recurrent neural network, has been used to model human lexical decision reaction times. Finally, our work is related to work on syntax that evaluated whether a word-level language model assigns a higher probability to an grammatical sentence than to a minimally different ungrammatical one (Lau et al., 2016; Linzen et al., 2016; Sennrich, 2017).

In summary, the main result of this study is that with a sufficient number of parameters character-level neural networks are able to perform lexical decisions with high levels of performance, despite not being trained on this task. A second important result is that the main predictor of lexical decision accuracy was the total number of parameters in the network; we found no evidence that deep networks are superior to shallow and wide ones on this task.

Acknowledgements

We thank Emmanuel Keuleers for his assistance with the Wuggy nonword generator. This research was supported by the European Research Council (grant ERC-2011-AdG 295810 BOOTPHON) and the Agence Nationale pour la Recherche (grants ANR-10-IDEX-0001-02 PSL and ANR-10-LABX-0087 IEC).

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

R. Harald Baayen, Petar Milin, Dusica F. Djurdjević, Peter Hendrix, and Marco Marelli. 2011. An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. *Psychological Review*, 118(3):438–481.

Alan Baddeley, Hazel Emslie, and Ian Nimmo-Smith. 1993. The spot-the-word test: A robust estimate of verbal intelligence based on lexical decision. *British Journal of Clinical Psychology*, 32(1):55–65.

David A. Balota, Melvin J. Yap, and Michael J. Cortese. 2006. Visual word recognition: The journey from features to meaning (a travel update). In *Handbook of psycholinguistics*, pages 285–375.

David. A. Balota, Melvin. J. Yap, Michael J. Cortese, Keith A. Hutchison, Brett Kessler, Bjorn Loftis, James H. Neely, Douglas L. Nelson, Greg B. Simpson, and Rebecca Treiman. 2007. The English lexicon project. *Behavior Research Methods*, 39(3):445–459.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703. Association for Computational Linguistics.

Ewan Dunbar, Gabriel Synnaeve, and Emmanuel Dupoux. 2015. Quantitative methods for comparing featural representations. In *Proceedings of the 18th International Congress of Phonetic Sciences*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*.

Emmanuel Keuleers and Marc Brysbaert. 2010. Wuggy: A multilingual pseudoword generator. *Behavior Research Methods*, 42(3):627–633.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749, Phoenix, AZ.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2016. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Scott A. McDonald and Richard C. Shillcock. 2001. Rethinking the word frequency effect: The neglected role of distributional information in lexical processing. *Language and Speech*, 44(3):295–323.

David E. Meyer and Roger W. Schvaneveldt. 1971. Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, 90(2):227–234.

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June. Association for Computational Linguistics.

Roger Ratcliff, Pablo Gomez, and Gail McKoon. 2004. A diffusion model account of the lexical decision task. *Psychological Review*, 111(1):159–182.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? Assessing MT quality with contrastive translation pairs. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

Ivelin Stoianov and John Nerbonne. 2000. Exploring phonotactics with simple recurrent networks. In Walter Daelemans, editor, *Proceedings of Computational Linguistics in the Netherlands, 2000*, volume 29, pages 51–68.

Ivelin Stoianov, Huub Bouma, and John Nerbonne. 1998. Modeling the phonotactic structure of natural language words with simple recurrent networks. In Hans van Halteren Peter-Arno Coppen and Lisanne Teunissen, editors, *Proceedings of Computational Linguistics in the Netherlands, 1997*, pages 77–95. Amsterdam: Rodopi.

Alberto Testolin, Ivelin Stoianov, Alessandro Sperduti, and Marco Zorzi. 2016. Learning orthographic structure with sequential generative neural networks. *Cognitive Science*, (3):579–606.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27.