Entity Decisions in Neural Language Modelling: Approaches and Problems

Jenny Kunz and **Christian Hardmeier**
Department of Linguistics and Philology
Uppsala University
752 36 Uppsala, Sweden
jenny.kunz.7402@student.uu.se
christian.hardmeier@lingfil.uu.se

Abstract

We explore different approaches to explicit entity modelling in language models (LM). We independently replicate two existing models in a controlled setup, introduce a simplified variant of one of the models and analyze their performance in direct comparison. Our results suggest that today's models are limited as several stochastic variables make learning difficult. We show that the most challenging point in the systems is the decision if the next token is an entity token. The low precision and recall for this variable will lead to severe cascading errors. Our own simplified approach dispenses with the need for latent variables and improves the performance in the entity yes/no decision. A standard well-tuned baseline RNN-LM with a larger number of hidden units outperforms all entity-enabled LMs in terms of perplexity.

1 Introduction

Reference to entities in the world is a core feature of human language, and coreference between different mentions in a text is a fundamental property of coherent communication. Computational approaches to reference have long been studied in the area of coreference resolution (Ng, 2017). Very recently, explicit models of reference have also been studied in the context of language modelling. The usual approach is to introduce latent variables modelling whether the next token is part of an entity mention, and which of the previously seen entities it refers to.

In this work, we present a comparative study of three language modelling approaches with explicit representations of entity coreference: YangLM, the entity-enabled language model (LM) of Yang et al. (2016), the EntityNLM model of Ji et al. (2017), and SetLM, our own extension of the latter. YangLM and EntityNLM differ in the parameterization of the latent variables and the order in which decisions are made. SetLM is a simpler architecture with fewer loss functions. It replaces the latent variable modelling the decision whether to produce an entity with two extra embeddings, one for a new entity (similar to the other models) and one for the case that the token does not belong to an entity. We replicate the results of Yang et al. (2016) and Ji et al. (2017) with an independent reimplementation of their models in a comparable experimental setup and evaluate the models in terms of overall language modelling performance performance in comparison with a simple RNN-LM. We also study the accuracy and precision/recall in each individual decision step and look at the convergence of variables. We find that YangLM outperforms the other models in terms of perplexity, whereas SetLM achieves the best results for the entity yes/no prediction. None of the entity-enabled LMs is competitive with a simple RNN-LM having a higher number of hidden units, and we do not achieve similar gains by enlarging the hidden sizes of the entity LMs.

2 Approaches

RNNs for language modelling have been state of the art for a few years (Mikolov et al., 2013), mostly using LSTMs (Hochreiter and Schmidhuber, 1997; Sundermeyer et al., 2012). They model each token in a document based on their previous context:

$$h_t = LSTM(x_t h_{t-1}). \quad (1)$$

The explicit incorporation of coreference in these LMs is a relatively new and less researched task. In the entity prediction process of such models, two fundamental decisions are made: (1) *Is the token (part of) an entity mention?* and (2) *Which entity does it refer to?* To our knowledge, Yang et al. (2016) were the first to implement this idea. In their model, (1) is handled with the variable z_t

for each position t with an attention mechanism (Bahdanau et al., 2014) with the LSTM hidden state over the set of observed entities h^e that also contains a learnable embedding e_{new} for a new entity that has not been observed yet. They estimate the probability distribution over the known entities and use the weighted sum for the decision whether the token is part of an entity mention.

$$p^{coref}(v_t|h^e, h_{t-1}) = ATTN(h^e, h_{t-1}) \quad (2)$$

$$d_t = \sum_{v_t} p(v_t) h^e_{v_t} \quad (3)$$

$$p(z_t|h_{t-1}) = sigmoid(W[h_{t-1}, d_t]) \quad (4)$$

If $z_t = 1$ (i.e. the word is an entity mention), the probability for the next word is calculated based on v_t (see Equation 2), the previous hidden state of the LSTM and the set h^e. If $z_t = 0$ then the next word is predicted based on the previous hidden state of the LSTM only.

EntityNLM (Ji et al., 2017) handles (1) with the variable R_t, corresponding to z_t, but in contrast to YangLM solely based on the LSTM hidden state, using a parametrized embedding associated with $r \in \{0, 1\}$. The decision which entity it refers to is handled with the variable $\mathrm{E_t}$ that denotes the index of the current entity in the set of known entities E_t in case $R_t = 1$, using the LSTM hidden state, the set of entities and a distance feature vector. The prediction of the next token x is always based on the LSTM hidden state h_{i1} and the representation of the current entity e even if it is not an entity. In this point EntityNLM also differs from YangLM that only uses the entity representation in the case that the token is predicted to be an entity token.

Ji et al. also introduce a length prediction variable L_t that is predicted when a new mention is started, using the last hidden state h_{t-1} and the most recent embedding of the entity e_t.

Clark et al. (2018) build on Ji et al. (2017) and track entities to use them as contextual information when generating narrative text. They evaluate their model in mention generation, sentence selection and sentence generation tasks, but not in the perplexity metric so that we cannot use their model for quantitative comparison.

Our SetLM model builds on YangLM and models each token in a document with an LSTM as above. It also saves the previously seen entities in a set, but instead of introducing a variable that controls if the next token is part of an entity mention or not, we include a learned embedding for the case that the token is not an entity token to the set, inspired by approaches in Question Answering with Answer Triggering (Zhao et al., 2017). The set E with the previously seen entities $e_1, ..., e_n$ has, besides e_{new} for the detection of a new entity, also contains the learnable embedding $e_{noentity}$ for the case that the token is not an entity. This makes it possible to dismiss the decision in Equation 4 in YangLM while keeping its remaining decision structure (Equation 1, 2 and 3).

The decision on the next token is, as in Yang et al.'s model, based on h_{t-1} and the corresponding embedding with the highest attention score in the set of entities if the token is part of a mention, and solely based on h_{t-1} otherwise.

3 Data

We train, optimize and evaluate the three models on the English subset of the OntoNotes 5.0 corpus (Weischedel et al., 2013) with 1.5 million words and anaphoric coreference annotation within a document, and use the CoNLL-2012 split into train, development and test set. We lowercased all tokens, replaced all numbers by a special symbol and all tokens with less than 5 occurrences with a special token for rare words, resulting in a vocabulary size of 11539. Like Ji et al. (2017), we keep only the embedding mentions where mentions are nested and removed all mention annotation where the mention length is higher than 25.

We did not reduce the length of the mentions in the data for YangLM to one as in the original model but use the setting as described above.

4 Implementation

We implemented all models in Python using the PyTorch deep learning library (Paszke et al., 2017). As candidate hyperparameters for the hidden size of the LSTM and word embedding layer, we tried the values 32, 48, 64, 128, 256, 512. We employ dropout (Srivastava et al., 2014) with candidate rates of 0.0, 0.1 or 0.2 and for the Adam optimizer (Kingma and Ba, 2014), we tried the learning rates 0.01, 0.005 and 0.001. We tried the models with GloVe (Pennington et al., 2014) and with randomly initialized, learnable word embeddings. We also experimented with a weighted loss with the intention to force the models to produce more entity mentions.

Based on the experimental results on the development set, we chose a hyperparameter setting for the model based on Yang et al. (2016) with 64 hidden units for both LSTM hidden size and word embedding size, Adam optimizer with $\lambda = 0.005$, a dropout rate of 0.2 and randomly initialized word embeddings. The model was trained for 20 epochs.

The best hyperparameter setting for the model based on EntityNLM was very similar and only differs in having a hidden size of 128 and being trained for 22 epochs. For SetLM, we chose a hidden size of 48 and 16 epochs.

For the evaluation of our main metric that is the token perplexity, we implement two baseline models that are purely LSTM-based LMs. We use the same architecture in two settings: one in the same hyperparameter setting as the best Yang et al. model with a hidden size of 64, trained for twelve epochs, and one optimized model with a hidden size of 512, trained for three epochs.

5 Evaluation

As the main metric for the language models general performance, we measure the perplexity. We also evaluate the entity prediction process qualitatively by measuring precision and recall for the question if the next token is part of an entity mention and the accuracy for the choice of the entity from the set, and evaluate the length prediction in the model based on EntityNLM. For our model, we regard the choice of $e_{noentity}$ as the *Entity No*-decision and the choice for either of the entities as the *Entity Yes*-decision. For the accuracy of the choice of the entity from the set, we only looked at the choices in the *Entity Yes*-case.

5.1 Perplexity

We report the results for our models and baselines on the test set along with the original results from Ji et al. (2017) in table 1.[1]

Based on these results, we cannot confirm that the models outperform a simple RNN-LM on the OntoNotes data set. Both RNN-LM baselines easily outperform both the re-implemented and the reported results of Ji et al. (2017), and the optimized baseline (RNN-512) also performs much better than the model based on Yang et al. (2016) and our model which though both outperform the RNN that has same hidden size as itself (RNN-64). The model based on Yang et al. clearly performs best among all entity-predicting models.

	All	**Ent.**	**Non-Ent.**
RNN-64	121	177	114
RNN-512	**96**	**126**	**88**
EntityNLM (rep.)	132	-	-
EntityNLM (own)	131	154	127
YangLM (own)	107	132	101
SetLM	114	154	108

Table 1: Token perplexity results

	Perplexity Ratio
RNN-64	0.68
RNN-512	0.76
YangLM (own)	0.81
EntityNLM (own)	**0.85**
SetLM	0.74

Table 2: Relation Perplexity All Tokens / Entity Tokens

The decision how to select a token seems to be generally harder on entity tokens in the data set as they generally and for all models have a higher perplexity than non-entity tokens. But measured by their overall performance, the entity tokens in the re-implemented EntityNLM and YangLM models are relatively better than in the other models, while SetLM lies between the baseline models. Table 2 shows the perplexity of all tokens divided by the perplexity of entity tokens only, giving a measure for the relative performance of the models on entity tokens. But these results must be seen with the constraint that EntityNLM and YangLM get access to gold entity lists, and that the perplexity is a metric that grows exponentially, which limits the comparability of the ratio. The fact that our model does not improve on entity tokens suggests that the improvements of the re-implemented EntityNLM and YangLM models are caused by the gold information.

5.2 Entity Prediction

As the models are optimized for perplexity, the following results would possibly have been better in other hyperparameter settings. We observed higher values on the development set during tuning and great oscillations of the scores for different

[1] Please note that we give the two re-implementations access to the correct entity lists at test time in order to be able to evaluate each of the decision steps independently. The original results by Ji et al. (2017) did not have this access to gold entity information, so that the results are not directly comparable. SetLM also comes without access to this information as the mention decisions are made in one step.

epochs which makes it hard to interpret specific results.

	Prec.	**Recall**	**F1**
YangLM (own)	**60.9**%	30.2%	40.3%
EntityNLM (own)	39.0%	53.9%	45.3%
SetLM	41.0%	**58.1**%	**48.1**%

Table 3: Entity Yes/No Prediction

Precision and recall of both models are low, suggesting that the question if the next word is an entity is highly challenging in a LM. SetLM has the highest F1 score, suggesting that the Entity Yes/No prediction is best handled without a discrete decision.

	Accuracy
YangLM (own)	65.8%
EntityNLM (own)	**67.8**%
SetLM	65.1%

Table 4: Entity Choice

The accuracy for the decision which entity to choose is comparatively high. The EntityNLM re-implementation obtains the best value with a substantial margin.

5.3 Length Prediction

The re-implementation of EntityNLM's length prediction is correct in 59.4% of all cases, with the average distance of the false predictions to the gold mention length being 2.85 tokens. The average lengths of the mentions in all three models differ only slightly, being 1.53 for the EntityNLM re-implementation, 1.43 for the Yang et al. re-implementation and 1.78 for SetLM. The average length in the Gold data is 2.25 tokens.

6 Discussion

While we cannot confirm that the incorporation of explicit entity information is helpful in a general language modelling task, and the models' abilities especially to predict where to form entities have shown to be limited, we see a potential for the continuous representations for entities to become better and to be useful in certain situations where entities re-appear over long distances, like in the narrative texts that were subject to Clark et al. (2018).

For short texts like in the OntoNotes data set, an RNN-based LM implicitly seems to learn enough information about entities and the error propagation caused by wrongly detected entities in the set and erroneous decisions in the prediction process outweigh the information gain compared to basing the decision on the hidden state only.

The models' results for the decision steps in the entity prediction process suggest that handling the question if the word belongs to an entity mention and which entity it is jointly with information from the set of entities is preferable over first deciding if the word belongs to an entity mention. The list of entities seems to be very helpful context for the question if the next word is an entity. As the question if the next token is an entity is by far the biggest error source, it will lead to relevant error propagation in real-world applications. Therefore and because the F1 score is best for SetLM we suggest that it is best handled implicitly. We note that with left-side context only, the decision if the next token belongs to an entity mention is extremely hard for a LM.

We suggest that mention length prediction is not crucial for a well-performing model. All systems tend to create shorter mentions than in the gold predictions to a similar extent and the EntityNLM re-implementation did not perform notably better than the other models without length prediction.

A main challenge in the models is the difficulty to find a good training setting. Unsatisfactorily, our models did not profit from more hidden units without pre-training, while a high number of hidden units was the modification that lead to the greatest performance boost for the baseline RNN-LM. We find it promising that with 64 hidden units, the Yang et al. re-implementation performs better than the RNN-LM, but this effect does not scale to larger hidden sizes.

7 Conclusion

Our evaluation of three LMs that explicitly model coreference decisions in comparison to standard RNN-LMs suggests that these procedures do not improve a LM in a general language modelling task in perplexity.

The overall performance and the entity prediction results suggest that the decision if the next token is an entity should be handled with a probability distribution over the set of entities rather than with the current hidden state alone.

We see a need for evaluations on larger high-quality annotated data sets to study if they can possibly improve the prediction process with left context only, and for evaluations on other genres. Short texts that are mostly news texts are probably not the genre that takes most profit of explicit entity information. It is possible that in longer texts or texts with complexly interacting characters that develop in the text, a language model would take greater profit from explicit entity modeling.

Despite the limitations of the current models, we regard it as worthwhile to invest in improvements, especially in the development of models that are less prone to error propagation, and to explore these models' potential.

Acknowledgements

This work was supported by the Swedish Research Council under grant 2017-930. We thank all anonymous reviewers for their constructive comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.

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

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. *arXiv preprint arXiv:1708.00781*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Vincent Ng. 2017. Machine learning for entity coreference resolution: A retrospective look at two decades of research. In *AAAI*, pages 4877–4884.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. OntoNotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *arXiv preprint arXiv:1611.01628*.

Jie Zhao, Yu Su, Ziyu Guan, and Huan Sun. 2017. An end-to-end deep framework for answer triggering with a novel group-level objective. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1276–1282.