Named Entity Recognition on military text using stacked Bi-LSTM CRF with multiheaded self-attention

Daniel Braun (310270790)[1] and Andrew Lee (470330979)[2]

[1] University of Sydney, NSW 2006, Australia
dbra3341@uni.sydney.edu.au
[2] University of Sydney, NSW 2006, Australia
alee9619@uni.sydney.edu.au

1 Data preprocessing

1.1 Word Embeddings

Modern NLP implementations typically use pre-trained word embeddings based on the methods Word2Vec [1], FastText [2, 3] or GloVe [4]. Whilst these methods capture sequence semantics by considering a word and its surrounding context, GloVe also considers the global co-occurrence between words in a corpus. GloVe's performance improvements on semantic-syntactic tests [4] makes up for its increased training time and memory consumption and thus is the chosen method. Among the pre-trained GloVe word embeddings in the Gensim library [5], the embeddings trained on a Wikipedia dataset were preferred to alternatives that were trained on tweets from Twitter. This is due to the formal nature of the provided dataset, consisting mostly of excerpts from news articles. Informative Wikipedia articles are more aligned syntactically and semantically to the prescribed dataset than the colloquial expressions typically associated with tweets. However, it is worth acknowledging that with greater computation resources, a custom GloVe embedding trained directly on new articles in the 2010s on Middle Eastern conflicts and warfare could have proven to be better than the pretrained options. To incorporate these pre-trained embeddings in the NER model, the weights of an embedding layer in the NER network were initalised with the weights of the pre-trained embeddings.

1.2 Defence Lexicon

The semantically rich vocabulary afforded by well chosen lexicons have long been used to assist the outcome of NLP tasks [6]. In the context of the prescribed dataset of military origin, there are several words that are used in the military and defence domain that are rarely used elsewhere (e.g. "guerilla", "demilitarized"). More importantly, there are many words whose semantics are different in a military setting than in other contexts (e.g. "intelligence", "shell"). In order to account for these differences, the pretrained word embeddings were reinforced with an extra binary element to indicate whether a particular word was a common military term. The “DOD Dictionary of Military and Associated Terms” [7] was chosen as a military corpus as it contained the definitions of hundreds of military terms. Figure 1 in the Appendix shows an extract of this document. Words from the entire document were extracted and stop words were removed using the spaCy library [8].

1.3 PoS Tags

PoS tagging classifies each token in a sentence to a particular class, such as nouns, pronouns, verbs, numbers, etc. Since all NER tags represent “entities”, they almost all correspond to types of nouns. The use of PoS tags in an NER model can therefore focus the model on the tokens that correspond to nouns. Moreover, the PoS tags can also aid the model by capturing the syntactic structure of a sentence. For example, a noun is often preceded by a determiner, and so having information about the location of determiners can help in locating potential entities. Indeed, PoS tags are used as features across NER literature [9] and even those outside the English language, as exemplified by Benajiba and Rosso [10] in an Arabic NER task (certain tokens in the prescribed dataset also contain characters of Arabic origin). SpaCy and its trained pipeline of statistical models is used to perform the PoS tagging on the dataset [8].

1.4 Dependency Tags

As opposed to PoS tagging, which classifies each individual word, *dependency parsing* is the process of finding the syntactic links between words in a sentence. A dependency parser links each word in a sentence with exactly one other word in the sentence in which it is dependent on, in an asymmetric relationship. An example dependency parsing is given in Figure 2 in the Appendix. More pertinent to NER, in any multiple-word entity there will always be a dependency link between two of the words in that entity. Therefore, by using dependency information as input to an NER model, it will theoretically be easier to recognise entities which consist of multiple words. Additionally, maintaining direct links to dependent words may also assist in recognising when an entity will begin or end. For example, a future timestamp will almost always be preceded by the word “at”. Jie, Muis, and Lu [11] even propose a NER model that is guided by a dependency parser to restrict the space of possible combination of entities in their semi-Markov CRF. The spaCy [8] package and its syntactic dependendency parser is leveraged again to obtain dependency tags for the dataset. To maximise the amount of dependency information used as input to the NER model, both the numerical embedding of the dependent word as well as the type of dependency relation are considered as embedding inputs.

1.5 ELMo Contextual Embeddings

Embeddings from Language Models (ELMo) create contextualised token representations through concatenating all hidden states of a trained biLSTM [12].

Intuitively, contextual embeddings could allow easier discrimination between tokens that are entities only in some contexts, and have been shown to improve performance in various NER tasks [13]. Additionally, ELMo embedding includes a convolution layer and highway networks over the characters allowing the model to form representation of out-of-vocabulary words. The pretrained ELMo model used for the task was obtained from [14].

2 NER model

The **Best** model proposed in this study contains a stacked Bi-LSTM layer, followed by a MHA layer and a CRF layer. The input embeddings used are a combination of "glove-wiki-gigaword-100" (size 100) [5] and pre-trained ELMo (size 768) embeddings [15], as well as a binary value indicating whether each word is part of the defence lexicon discussed in Section 1.2. These embeddings initialise a trainable embedding layer in the NER network. Concatenated to the output of this embedding layer, is a vector of PoS tags corresponding to the words in the input sentence. Dropout is applied to this combined tensor, before being passed to a stacked Bi-LSTM which contains Dropout on each layer except the last. The outputs of the Bi-LSTM are then passed to a multi-head attention (MHA) layer. Dropout is again applied to the output of the MHA, before leading to a dense layer that outputs a tensor the same size as the tagset. Finally, this is passed through the CRF to obtain NER tag predictions. In short, Best contains ELMo+GloVe+PoS+Defence+Bi-LSTM+MHA+CRF with Dropout layers throughout.

2.1 Stacked Seq2Seq model

Recurrent Neural Networks (RNN) have long been applied to sequence to sequence (Seq2Seq) tasks like NER [16] to preserve historical text features vital in contextualising the prediction outputs. Over the years, long short-term memory (LSTM) [17] and gated recurrent units (GRU) [18] have been proposed to address the shortcomings of the vanilla RNNs. Vanilla RNNs are known to suffer from exploding and vanishing gradient problems, impacting their ability to capture long term dependencies [19]. LSTMs make use of 3 gates (forget, input, output) to mitigate the effects of the vanishing gradient problem [20] whilst the less complex GRU does so with 2 gates (reset and update). In this particular NER task where there is a limited amount of data, the faster training offered by GRUs (relative to LSTMs) is less important. Therefore, a *stacked bidirectional LSTM* (Figure 4 in the Appendix) is chosen to encode the sequential information in the input sentence. The inputs to the first layer of the bi-LSTM are the input embeddings from Section **??**. These inputs are then passed through two sequences of LSTM cells (Figure 3 in the Appendix), one passing forward through the sentence, and the other passing backwards. The use of a bi-directional LSTM allows information from past and future textual contexts to be considered at each time step and has been shown to empirically outperform an uni-directional LSTM

[16]. The output to the first layer consists of the concatenation of hidden states from the two LSTM cells corresponding to that word (one from the forward pass and one from the backward pass). These outputs are then used as inputs to an identical bi-LSTM, and this process is repeated for each extra layer in the stacked bi-LSTM. The optimal choice of the number of layers in the stacked bi-LSTM, as well as the size of the hidden state in each LSTM cell, is heavily dependent on the task and data. The selection of these parameters is detailed in Section 3.

2.2 Attention

Attention was originally proposed for Seq2Seq tasks to overcome the inability of the single context vector to remember long sequences [21]. Attention creates a direct connection between the context vector and the encoder states to focus on the relevant parts of the input sequence and has been shown to help with the vanishing gradient problems and bottlenecks [22]. More generally, attention can be defined as a technique used to compute a weighted sum of key-value pairs, dependent on a query. For keys (k_i), values (v_i) and query (q), the attention is given by the following expression $A(q,k) = \sum_{i=1}^{N} SoftmaxScore(q,k_i) \cdot v_i$. There are different ways of computing attention scores before softmaxing, below are 3 commonly used attention scores used in the research. The choice of these 3 allows the exploration of two fundamentally different alignment concepts in the dot product or cosine and offers insight into whether either is more appropriate for the NER task. The choice of the scaled dot-product also included to investigate whether the theoretical improvements of the scaling factor [23] is beneficial in a situation when the dimensionality in some of the proposed experiments.

- Dot-Product [24]: $A(q,k_i) = q^T k_i$
- Scaled Dot-Product [23]: $A(q,k_i) = \frac{q^T k_i}{\sqrt{n}}$
- Content-base [25]: $A(q,k_i) = cosine[q,k_i]$

Recently, transformers have popularised the use of MHA, which extends the use of attention beyond RNN decoders through self-attention mechanisms where each element in sequence attends to every other element in the sequence [23]. This addresses certain issues with the RNNs such as the lack of parallelisability and serves as an essential constituent of state of the arts outcome across many sequencing tasks [26]. To make use of this for the NER task, this project applies a MHA layer to the hidden states of the stacked bi-LSTM model, which serves as the queries (Q), keys (K), values (V). The choice of this position to place the attention revolves around the fact that bi-LSTM states are condensed local contextual summaries which should be "globalised" via attention to add another set of enriching the features to the next layer. Learnable weight matricies $W^Q_{1,\ldots h}$, $W^K_{1,\ldots h}$ and $W^V_{1,\ldots h}$ are used to transform Q,K,V into h sub-queries, sub-keys and sub-values (h different heads). These results have self-attention scores computed independently and concatenated, after which a final weight matrix W^O is applied to obtain a final MHA output.

$$MHA(Q, K, V) = \underset{i \in \{1 \ldots h\}}{\text{Concat}}(A(QW_i^Q, KW_i^K, VW_i^V))W^O \quad (1)$$

2.3 Conditional Random Fields

After MHA output is computed, it is concatenated with the original bi-LSTM outputs and fed to a trainable weight matrix that is used to project the results into the entity space ($P \in \mathbb{R}^{n \times k}$) for a sentence with n words and k entity classes. P can be treated as the score for each label and fed to a Conditional Random Field (CRF) which can handle the dependency between adjacent predicted entities [27]. Formally, the CRF computes its own score for each sequence of tags ($y_{1:n}$) using a transition matrix and P. These CRF scores can then be used by decoding algorithms such the Viterbi algorithm which uses dynamic programming to obtain the sequence of entities with the maximum score [28]. CRF has been shown to improve results in various NER tasks [10, 29] so it has been included in the best model.

3 Evaluation

3.1 Evaluation setup

Dataset The labelled NER data provided comes from a subset of the *re3d* [30] dataset. This subset is split in to a training set of size 564, a validation set of size 191, and a test set (without labels) of size 195. In the model evaluation, all models were trained on the training set, and subsequently evaluated on the validation set. In the NER problem, each token is assigned to exactly one label, and thus the mean F1 score and accuracy are equivalent. Given this equivalence, only accuracy on the validation set is presented as the comparison metric.

Baselines Several baselines were created in order to compare the addition and removal of particular features of the proposed model:

– **Baseline 1 (Single-layer Bi-LSTM+CRF)**: The input embeddings pass through a single-layer Bi-LSTM, and then pass directly to the CRF layer.
– **Baseline 2 (Bi-LSTM+CRF+Dropout)**: Same as Baseline 1 but with a stacked Bi-LSTM including Dropout layers, and a Dropout layer after the input embedding layer.
– **Baseline 3 (Bi-LSTM+Single-Head Attention+CRF+Dropout)**: Same as Baseline 2 but with a single-head attention layer (including Dropout) between the Bi-LSTM and the CRF.
– **Baseline 4 (Single-layer Bi-LSTM+MHA+CRF+Dropout)**: A single-layer Bi-LSTM followed by MHA and CRF layers, with Dropout throughout.

- **Baseline 5 (GloVe+Bi-LSTM+MHA+CRF+Dropout)**: Same as Baseline 4 but with stacked Bi-LSTM layers and only a GloVe embedding (no ELMo or PoS or defence lexicon).

As is the case for the Best model, all baselines apart from Baseline 5 use a concatenation of GloVe and ELMo input embeddings, as well as PoS tags, and a defence lexicon membership binary value.

Implementation With regards to hyperparemeter settings: the Best model contains a Bi-LSTM hidden dimension size of 300; the stacked Bi-LSTM contains 3 layers; the learning rate is set to 0.01 using stochastic gradient descent; weight decay set to 0.0001; a dropout value of 0.5 is used before and during the Bi-LSTM layers, and after the attention layer; the models were run on 100 epochs, with the best validation accuracy reported; 3 heads in the multi-head attention layer are used. While the model infrastructure was based mostly on theoretical ideas from the literature, the hyperparameter values were set based on the experimental results in Section 3.2.

Where applicable, each baseline model uses the same hyperparameters as the Best model.

3.2 Evaluation result

To illustrate the overall performance of the best model, its accuracy on the validation set is compared to that of Baseline 1 in Table 1. The Best model improves on the baseline by more than 2%, a significant increase in the context of this dataset.

Model	Acc
Baseline 1	83.64
Best	**86.22**

Table 1: Best model performance compared to Baseline 1 (Single-layer Bi-LSTM+CRF) on the validation set.

To evaluate the effect of using different input embeddings, an experiment was performed which added semantic and syntactic input features to Baseline 5 (containing only a GloVe input embedding). Table 2 contains the results of this experiment. It can be seen that, when added individually to Baseline 5, ELMo embeddings, and to a much lesser extent, PoS tags, both improve the model. However, the Best model which contains a combination of GloVe, PoS, Defence and ELMo, gives a better performance than each addition alone, or the combination of additions (i.e. *+All*).

EmbedType	Acc
Baseline 5	84.55
+PoS	84.64
+Dep	83.96
+Defence	84.51
+ELMo	85.68
+All	84.98
Best	**86.22**

Table 2: Comparison of input embeddings to Baseline 5 (only GloVe input embeddings). *Dep* includes the index of the dependent word, and its word embedding. *Defence* represents membership to the defence lexicon. ELMo indicates the concatenation of ELMo with GloVe.

The variants of attention output discussed in Section 2.2 were compared empirically with relation to Baseline 2 (Bi-LSTM+CRF+Dropout). The results are presented in Table 3. As can be seen from the table, all forms of attention improve model performance, with the "cosine" variant having the highest accuracy on the validation set[3].

Strategy	Acc
Baseline 2 (No attn)	85.89
cosine	**86.37**
Best (scaled)	86.22
dot	85.82

Table 3: Comparing variants of attention output.

Given that the best model uses multi-head attention, an empirical evaluation of the optimal number of heads to use was undertaken. These are compared to Baseline 3 (Bi-LSTM+Single-Head Attention+CRF+Dropout) in Table 4. While the validation accuracy is noisy with respect to the number of heads used, using 3 heads leads to a maximum validation accuracy.

The optimal number of Bi-LSTM stacked layers for this experiment was evaluated and presented in Table 5. Compared to Baseline 4, utilising two or three

[3] Note that the "cosine" accuracy is slightly higher than the Best (scaled) accuracy. An exhaustive grid search over many possible hyperparameter values may have found this attention strategy as a part of the Best model configuration. Given the number of hyperparameters and the computational time of running such experiments, sub-optimal results such as these are difficult to avoid.

NumHeads	Acc
Baseline 3 (1 head)	86.12
Best (3 heads)	**86.22**
6	85.97
10	85.80
16	85.63

Table 4: Evaluation of the optimal number of heads in multi-head attention.

stacked layers improves the performance of the model. However, this performance increase plateaus at three layers, and begins declining.

NumLayers	Acc
Baseline 4 (1 layer)	85.38
2	**86.23**
Best (3 layers)	86.22
4	85.31
5	85.31

Table 5: Evaluation of the optimal number of layers in the stacked Bi-LSTM.

To evaluate the value of CRF, an ablation study was undertaken which compares the performance of the best model with a model that removes CRF entirely. The final predictions of this ablated network were obtained by taking the argmax after the final linear layer in the network (after the attention layer). The results of this study are presented in Table 6, where it can be seen that CRF is a valuable feature of the model.

Strategy	Acc
Best without CRF	80.87
Best with CRF	**86.22**

Table 6: Evaluation of the optimal number of layers in the stacked Bi-LSTM.

To determine the optimal hidden layer size of the Bi-LSTM, various values were tested and compared to the best model with a hidden size of 300. The results in Table 7 of the Appendix show that the larger hidden layer sizes of 300 and 600 perform better than smaller embeddings, with the accuracy reaching a maximum at 300.

Table 8 in the Appendix shows that dropout improves the validation accuracy by more than 2%, indicating the importance of regularisation on the relatively small dataset. Note that value of 0.5 was chosen based on its theoretical foundations (it induces the highest variance for the output distribution of the layer), and its empirical efficacy in works such as [31].

References

[1] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[2] Piotr Bojanowski et al. "Enriching word vectors with subword information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[3] Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[5] Radim Rehurek and Petr Sojka. "Gensim–python framework for vector space modelling". In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011).

[6] Maite Taboada et al. "Lexicon-based methods for sentiment analysis". In: *Computational linguistics* 37.2 (2011), pp. 267–307.

[7] *Dictionary of Military and Associated Terms*. URL: `https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/dictionary.pdf`.

[8] Matthew Honnibal et al. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: `10.5281/zenodo.1212303`. URL: `https://doi.org/10.5281/zenodo.1212303`.

[9] Ling Luo et al. "An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition". In: *Bioinformatics* 34.8 (2018), pp. 1381–1388.

[10] Yassine Benajiba and Paolo Rosso. "Arabic named entity recognition using conditional random fields". In: *Proc. of Workshop on HLT & NLP within the Arabic World, LREC*. Vol. 8. Citeseer. 2008, pp. 143–153.

[11] Zhanming Jie, Aldrian Muis, and Wei Lu. "Efficient dependency-guided named entity recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[12] Matthew E Peters et al. "Deep contextualized word representations". In: *arXiv preprint arXiv:1802.05365* (2018).

[13] Matej Ulčar and Marko Robnik-Šikonja. "High quality ELMo embeddings for seven less-resourced languages". In: *arXiv preprint arXiv:1911.10049* (2019).

[14] Matt Gardner et al. "AllenNLP: A Deep Semantic Natural Language Processing Platform". In: 2017. eprint: `arXiv:1803.07640`.

[15] Ciprian Chelba et al. "One billion word benchmark for measuring progress in statistical language modeling". In: *arXiv preprint arXiv:1312.3005* (2013).

[16] Zhiheng Huang, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging". In: *arXiv preprint arXiv:1508.01991* (2015).

[17] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[18] Kyunghyun Cho et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[19] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[20] Sepp Hochreiter and Jürgen Schmidhuber. "LSTM can solve hard long time lag problems". In: *Advances in neural information processing systems* (1997), pp. 473–479.

[21] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[22] Ying Sha and May D Wang. "Interpretable predictions of clinical outcomes with an attention-based recurrent neural network". In: *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. 2017, pp. 233–240.

[23] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015).

[25] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural turing machines". In: *arXiv preprint arXiv:1410.5401* (2014).

[26] Andrea Galassi, Marco Lippi, and Paolo Torroni. "Attention in natural language processing". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[27] John Lafferty, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: (2001).

[28] Andrew Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE transactions on Information Theory* 13.2 (1967), pp. 260–269.

[29] Burr Settles. "Biomedical named entity recognition using conditional random fields and rich feature sets". In: *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications (NLPBA/BioNLP)*. 2004, pp. 107–110.

[30] Defence Science and UK Technology Laboratory. "Relationship and Entity Extraction Evaluation Dataset". In: (2017). URL: `https://github.com/dstl/re3d`.

[31] Pierre Baldi and Peter J Sadowski. "Understanding dropout". In: *Advances in neural information processing systems* 26 (2013), pp. 2814–2822.
[32] Caren Han. *Dependency Parsing Example, COMP5046 Lecture7 2021.*
[33] JM Zhou. *LSTM Cell.* URL: `https://stackoverflow.com/questions/50488427/what-is-the-architecture-behind-the-keras-lstm-cell`.
[34] COMP5046 Organisers. *Stacked Bi-LSTM.* URL: `https://www.kaggle.com/c/2021-comp5046-a2/overview/ner-model`.

4 Appendix

4.1 Supplementary Evaluation Results

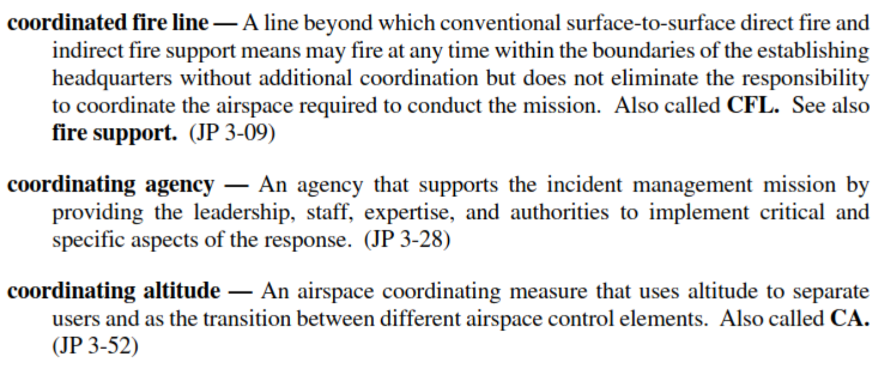
HiddenSize	Acc
50	84.93
100	85.32
Best (300)	**86.22**
600	85.87

Table 7: Evaluation of the optimal Bi-LSTM hidden size.

Dropout	Acc
Best without dropout	84.15
Best with dropout=0.5	**86.22**

Table 8: Effectiveness of Dropout on the best model.

4.2 Model Components



coordinated fire line — A line beyond which conventional surface-to-surface direct fire and indirect fire support means may fire at any time within the boundaries of the establishing headquarters without additional coordination but does not eliminate the responsibility to coordinate the airspace required to conduct the mission. Also called **CFL.** See also **fire support.** (JP 3-09)

coordinating agency — An agency that supports the incident management mission by providing the leadership, staff, expertise, and authorities to implement critical and specific aspects of the response. (JP 3-28)

coordinating altitude — An airspace coordinating measure that uses altitude to separate users and as the transition between different airspace control elements. Also called **CA.** (JP 3-52)

Fig. 1: An extract from "DOD Dictionary of Military and Associated Terms" [7].

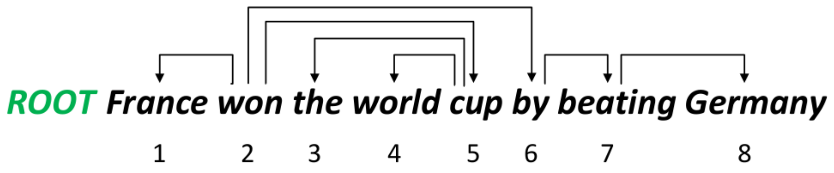


Fig. 2: Dependency parsing of a sentence [32].

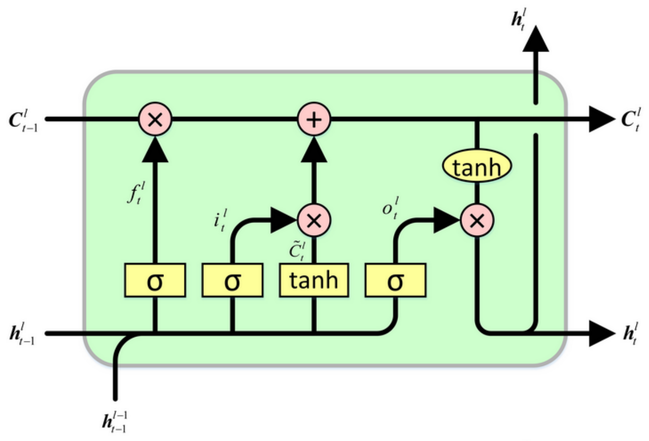


Fig. 3: Single LSTM cell [33]

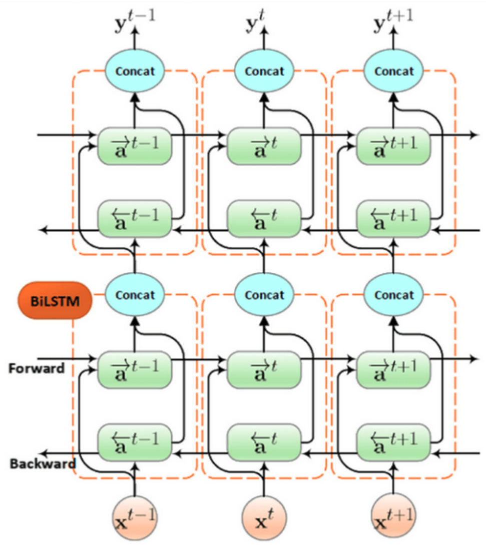

$\mathbf{y}^{t-1}$
$\mathbf{y}^{t}$
$\mathbf{y}^{t+1}$
Concat
$\overrightarrow{\mathbf{a}}^{t-1}$
$\overrightarrow{\mathbf{a}}^{t}$
$\overrightarrow{\mathbf{a}}^{t+1}$
$\overleftarrow{\mathbf{a}}^{t-1}$
$\overleftarrow{\mathbf{a}}^{t}$
$\overleftarrow{\mathbf{a}}^{t+1}$
BiLSTM
Forward
Backward
$\mathbf{x}^{t-1}$
$\mathbf{x}^{t}$
$\mathbf{x}^{t+1}$

Fig. 4: Stacked bi-LSTM [34]