Attending to Characters in Neural Sequence Labeling Models

Marek Rei, Gamal K.O. Crichton, Sampo Pyysalo

University of Cambridge

Sequence Labeling

The task:

Given a sequence of tokens, predict a label for every token.

POS-tagging:

```
DT  NN    VBD      NNS    IN      DT  NN     .
The pound extended losses against the dollar .
```

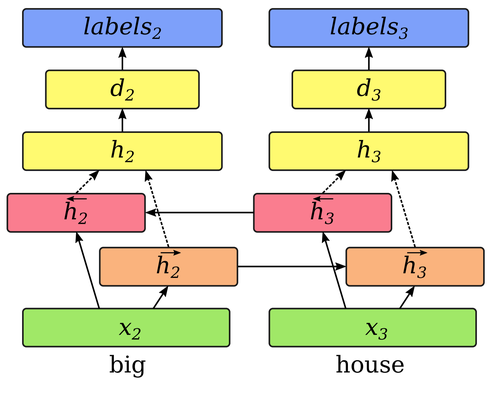
Named Entity Recognition:

```
PER _     _   _      _   ORG ORG   _   TIME _
Jim bought 300 shares of Acme Corp. in 2006 .
```

Error Detection:

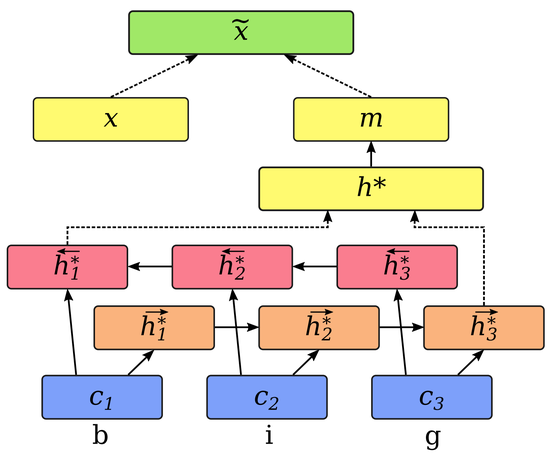
```
+ +   +  -      +  +     +  +   -     +
I like to playing the guitar and sing louder .
```

Bidirectional Word-level LSTM



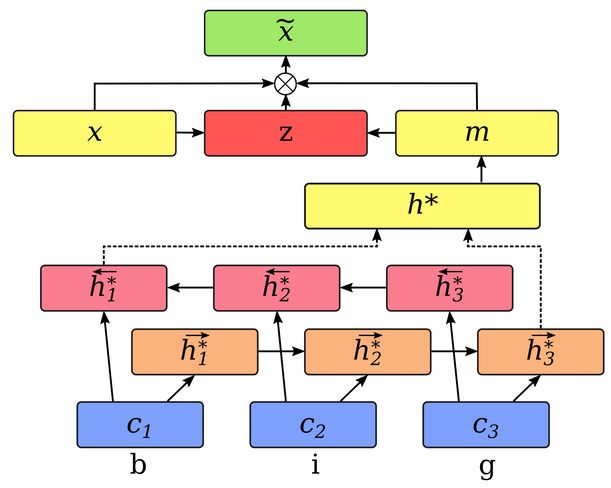
- **Sequence of tokens** mapped to word embeddings.
- **Bidirectional LSTM** builds context-dependent representations for each word.
- A small **feedforward layer** encourages generalisation.
- **Conditional Random Field** (CRF) at the top outputs the most optimal label sequence for the sentence.
- Unable to model **unseen words**, learns poor representations for infrequent words, and unable to capture character-level patterns.

Concatenating Character Representations



- **Character embeddings** are used to represent each letter.
- **Bidirectional LSTM** builds a representation m for the word.
- The character-level representation can be combined with the word embedding via **concatenation**: $\tilde{x} = [x; m]$.
- The resulting vector $\tilde{x}$ is used as the word representation in the **sequence labeling** model.
- Both the character-level component and the word-level sequence labeling model are **trained together**.

Attending to Character Representations



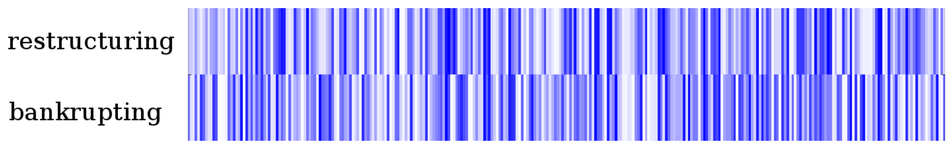
- **Character-based representation** m built using a bidirectional LSTM.
- A **vector of weights** z is dynamically predicted based on x and m.
- Word representations x and m combined using a **weighted sum**: $\tilde{x} = z \cdot x + (1 - z) \cdot m$
- Extra loss term optimises **m to be similar to x**, but not x to be similar to m:

$$\widetilde{E} = E + \sum_{t=1}^{T} g_t(1 - cos(m^{(t)}, x_t)) \qquad g_t = \begin{cases} 0, & \text{if } w_t = OOV \\ 1, & \text{otherwise} \end{cases}$$

- The model can **dynamically decide** whether to take each feature from the character-based representation or the designated word embedding.

Analysis

- **Visualisation** of attention weights for two words, trained on the PTB-POS dataset.
- Darker blue indicates features with **higher weights** for the representation built from characters.
- The model **actively chooses** to assign high weights to the character-based representation.



- The patterns are different, indicating that the choices are indeed being made **dynamically** for each word.

Conclusion

- **Word embeddings** map similar words to similar vectors, but they learn poor representations for infrequent words, and do not take advantage of character patterns.
- Integrating **character-level features** into sequence labeling increased performance on all benchmarks.
- **Dynamically combining** character-based and word representations consistently outperformed concatenation, even using fewer parameters.

Results

- Experiments on **8 different datasets** and 4 different tasks: POS-tagging, named entity recognition, error detection, and chunking.

	CoNLL00		CoNLL03		PTB-POS		FCEPUBLIC		BC2GM		CHEMDNER		JNLPBA		GENIA-POS	
	DEV	**TEST**	**DEV**	**TEST**	**DEV**	**TEST**	**DEV**	**TEST**	**DEV**	**TEST**	**DEV**	**TEST**	**DEV**	**TEST**	**DEV**	**TEST**
Word-based	91.48	91.23	86.89	79.86	96.29	96.42	46.58	41.24	84.07	84.21	78.63	79.74	75.46	70.75	97.55	97.39
Char concat	92.57	92.35	89.81	83.37	97.20	97.22	46.44	41.27	87.54	87.75	82.80	83.56	76.82	72.24	98.59	98.49
Char attention	**92.92**	**92.67**	**89.91**	**84.09**	**97.22**	**97.27**	**47.17**	**41.88**	**87.98**	**87.99**	**83.75**	**84.53**	**77.38**	**72.70**	**98.67**	**98.60**