CONTEXT-BASED

SENTENCE SEGMENTATION

FOR IRREGULAR DOMAINS

Bachelor’s Thesis
Krisztina Agoston
Problem definition

Approaches

Machine Learning Solution

Evaluation

Conclusion
NLP pipeline

NLP = Natural Language Processing

Sentence segmentation:

- find the sentence boundaries in the input text
- divide the input text into individual sentences
- Formal definition:
  map the input text \( x \) to \( y_1 \ldots y_n \) sentences
LORANSTA Marcus Island was billeted for 23 U.S. Coast Guard personnel. The commissioning commanding officer was U.S. Coast Guard Lieutenant Commander Louis. C. Snell.
Complexity

Ambiguity examples:

- Abbreviation: *U.S. Special Operations Forces*
- Abbreviation at sentence end: *in Washington D.C.*
- Initials: *J. F. Kennedy*
- Ordinal numbers: *1. Section*
- Ellipses: *I have a dream....I have a dream today.*
- Quotes: *[…] 24 years later in his ”Tear down this wall!” speech.*

How many punctuations do not denote a sentence end? It depends...

- Brown: 12%
- Wall Street Journal: 42%
Problem definition

Approaches

Machine Learning Solution

Evaluation

Conclusion
Approaches for sentence segmentation

Rule based
  ◦ Find sentence boundaries with predefined rules

Statistical
  ◦ Collect statistics from the text
  ◦ Calculate probabilities to disambiguate punctuation

Machine learning
  ◦ Use neural networks
  ◦ Train the automated system
• Machine learning in a Nutshell

Based on neural networks

Neurons (nodes) build up a network

Each neuron calculates \( y = \sigma(\sum(w_i \cdot x_i) + b) \)

Goal: given the input \( x_1 \ldots x_n \) predict the output \( y \)

- \( i \) is the index of the sample
- \( w_i \in \mathbb{R} \) is some weight to determine the input’s importance
- bias \( b \in \mathbb{R} \) is a constant value
- Nonlinear activation function \( f \) for creating probability \( \sigma(z) = \frac{1}{1 + e^{-z}} \) for the classes
Problem definition

Approaches

Machine Learning Solution

Evaluation

Conclusion
Long Short-Term Memory Neural Network (LSTM)

- preferred to be used for sequential data
- capable of learning long-term dependencies

- distinguished by its gated structure:
  - Forget gate $f_f$
  - Input gate $f_i$
  - New candidate $f_{nc}$
  - Output gate $f_o$
- and its cell state $c_t$
  - $= \text{memory}$

I was born in France and lived there for 10 years. I can speak French.
Context windows

Character- or token-based

- For every unit in the text
- For every punctuation
- For every possible sentence end marks

The 35th U.S. President J. F. Kennedy died in 1963.

Window size = 6:

```
<table>
<thead>
<tr>
<th>the</th>
<th>35th</th>
<th>u</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>died</th>
<th>in</th>
<th>1963</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Labels:

0

1
Layers in the LSTM model / 1

- **Input layer** provides data in the correct form
  - context windows in batches

- **Embedding layer** encodes the units
  - Creating embedding vectors

- **LSTM layer**
  - perform a non-linear computation for each unit in a batch

- **Linear layer**
  - transforms the output of the hidden layer to the label space

\[ y = xA + b \]

\[ \text{softmax}(\vec{v}) \]

- **Softmax layer**
  - threshold

- **Output layer**
  - Labels

- **Raw text**
  - 128 LSTM neurons
Layers in the LSTM model / 2

Softmax layer
- transforms the real numbers within a vector into real numbers between 0 and 1 so that they sum up to 1
  \[ \text{softmax}(v) = \frac{e^{v_i}}{\sum_{i=1}^{n} (e^{v_i})} \]
- We can use these values as probabilities

Output layer
- Label \[ \begin{cases} 1 & \text{if probability} \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases} \]
Bidirectional-LSTM hidden layer

Backward output

Forward output

middle

LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM

LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM

x1
x2
x3
x4
x5
x6
x7
- Wall Street Journal (WSJ)
  - Sentences from the journal

- Brown
  - Works published in the USA in 1961

- Europarl
  - Extracted from the proceedings of the European Parliament

- Wikipedia
  - Paragraphs from Wikipedia entries

- arXiv
  - Paragraphs from scholarly articles from several research domains

Irregular example:

Datasets

Divided: 80% for training

- 10% for development
- 10% for final test

Training set from paragraphs:

- Take paragraphs shorter than 100 characters
- split paragraphs on punctuation + whitespace + sentence starter (however, because, the, she)
- Errors: sentence starts with a word not in our set punctuation followed by word from our set (not significant)

<table>
<thead>
<tr>
<th></th>
<th>sentences</th>
<th>paragraphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSJ</td>
<td>3.914</td>
<td></td>
</tr>
<tr>
<td>Brown</td>
<td>56.323</td>
<td></td>
</tr>
<tr>
<td>Europarl</td>
<td>1.906.966</td>
<td></td>
</tr>
<tr>
<td>arXiv</td>
<td>1.006.228</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>45.676.715</td>
<td></td>
</tr>
</tbody>
</table>
Test set for evaluation

Hard cases from the result of NLTK and spaCy segmentizers

Random cases

Size of the test subset with random sentences and hard cases

<table>
<thead>
<tr>
<th>corpus</th>
<th>hard cases</th>
<th></th>
<th>random</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>paragraphs</td>
<td>sentences</td>
<td>paragraphs</td>
<td>sentences</td>
</tr>
<tr>
<td>Wall Street Journal</td>
<td>571</td>
<td>3.914</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Europarl</td>
<td>1043</td>
<td>6.876</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brown</td>
<td>1.409</td>
<td>9995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arXiv</td>
<td>1.000</td>
<td>4.091</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>775</td>
<td>2.928</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>929</td>
<td>3.267</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Metrics

Error rate = \( \frac{FP + FN}{FP + FN + TP} \)

Precision = \( \frac{TP}{FP + TP} \)

Recall = \( \frac{TP}{FN + TP} \)

F1-score = \( \frac{TP}{TP + \frac{1}{2}(FP + FN)} \)

Accuracy = \( \frac{\text{Correct sentences}}{\text{All sentences}} \)

Confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Positive</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Actual Negative</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

True positive: correctly predicted boundary

False positive: no actual sentence boundary predicted as one

False negative: missed sentence boundary

True negative: most of the cases not a sentence boundary, not predicted as a sentence boundary
Baseline systems

Baseline algorithm
- Define a lower bound
- Split on punctuation + whitespace + capitalized letter
  \r"([\./\?\!\"\)])\s+([A-Z])"

NLTK
- Complete NLP toolkit
- Statistical segmentizer: Punkt sentence tokenizer
- Trainable with unsupervised training

spaCy
- State-of-the-art system with diverse components
- Senter trainable neural network-based segmentizer
Accuracy by growing context size

Character-based models

Token-based models

- The larger context improves accuracy values by both LSTM models
- Over the optimum the uni-LSTM’s accuracy drops
Model trained on Wikipedia corpus, evaluated on the development set (2753 sentences with hard cases)
Small dataset trained for 100 epochs, large for 10 epochs

Larger training data improves accuracy, especially by LSTM
Character-based models perform better
Comparison of accuracy of different systems on various corpora

Uni- and bi-STM models: character-based context window on punctuation, trained with early stopping

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Baseline</th>
<th>LSTM(9)</th>
<th>bi-LSTM(15)</th>
<th>default</th>
<th>trained</th>
<th>senter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europarl</td>
<td>90.98</td>
<td>95.72</td>
<td>96.80*</td>
<td>95.39</td>
<td>95.68</td>
<td>95.10</td>
</tr>
<tr>
<td>arXiv hard</td>
<td>92.30</td>
<td>90.37</td>
<td>92.91</td>
<td>74.31</td>
<td>87.29</td>
<td>91.93</td>
</tr>
<tr>
<td>Brown</td>
<td>67.43</td>
<td>92.17</td>
<td>93.94*</td>
<td>85.01</td>
<td>83.91</td>
<td>93.84</td>
</tr>
<tr>
<td>Wikipedia hard</td>
<td>80.12</td>
<td>82.04</td>
<td>90.64*</td>
<td>84.19</td>
<td>89.07</td>
<td>79.44</td>
</tr>
<tr>
<td>Wikipedia rand.</td>
<td>91.86</td>
<td>93.44</td>
<td>96.94</td>
<td>99.42</td>
<td>97.86</td>
<td>95.56</td>
</tr>
<tr>
<td>WSJ</td>
<td>69.79</td>
<td>72.75</td>
<td>89.29</td>
<td>92.79</td>
<td>91.74</td>
<td>80.32</td>
</tr>
</tbody>
</table>

* bi-LSTM was trained on 500,000 sentences

Bi-LSTM model outperforms other systems on most of the corpora
Cross-corpora evaluation: trained on Wikipedia, evaluated on WSJ
Review

Automated, trainable, easily adaptable solution for SBD

Deal with sequential data

Involve not only the previous inputs in the current calculation, but also the upcoming context

Irregular domains

→ Neural Networks (NN)

→ Special kind of NN: LSTM

→ Bidirectional LSTM

→ arXiv, Wikipedia
Conclusion

Approach:
- Character-based approach improves recall and decreases error rate

Context
- Using previous and next context improves precision by bi-LSTM, that outperforms other systems

Training:
- Large training data to get an accurate model

Domains:
- The type of the domain determines the task-complexity

Goal of future work: general purpose model with improved runtime
Sources

Extra slides

**Rule-based approach**
**Statistical approach**
**Dependencies**
**Embeddings**
**Data preparation**
**Context window** 1
**Context window** 2
**F1 score**
**FB score**

**Baseline performance**
**Training time - context size**
**Systems on Wikipedia**
**Hard cases vs. random**

**Dimensionalities**
**Threshold**
**Training setup**
**Early stopping**
**Bi-LSTM**
Common Errors - uni LSTM vs. Bi-LSTM

Unidirectional LSTM:

several punctuation marks next to each other

◦ It was as if he didn’t want the guests to be there. || ”
◦ Give the enemy no rest . ||. Do all the

Period after a number

◦ or VBLANK interrupts on IRQ 5. This allows the use

Bidirectional LSTM

split after semicolon

◦ Sergio Ortega; || Leon Schidlowsky;
Common errors

False negative predictions are the missed sentence boundaries:
- .. translated as "Qīng Chéng" () The name...
- at all hazards. ... Give the enemy no rest ...
- "the sexiest man on TV." As Eddie films
- in Washington D.C. Before and

False positive ||
- "chenel" || "canal".
- "Do You Miss Me?" || became a Top 40 hit
- including Yahoo! Wireless in London, Splash! || in Germany,
Character based bi-LSTM(15) model evaluated on different corpora

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Error rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>arXiv hard</td>
<td>0.05</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>92.91%</td>
</tr>
<tr>
<td>Europarl</td>
<td>0.02</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>96.80%</td>
</tr>
<tr>
<td>Wikipedia hard</td>
<td>0.08</td>
<td>0.94</td>
<td>0.97</td>
<td>0.96</td>
<td>90.64%</td>
</tr>
<tr>
<td>Brown</td>
<td>0.03</td>
<td>0.99</td>
<td>0.97</td>
<td>0.98</td>
<td>93.94%</td>
</tr>
</tbody>
</table>

Character-based bi-LSTM model, Context window size is 15 trained with the early stopping method on the corresponding corpus

- Wikipedia hard has lower precision because in text set we do not split on semicolon, but in training set there are many such cases
Compare systems on the Wikipedia corpus

Character-based models, context for sentence ending punctuations, trained on 500,000 Wikipedia sentences with early stopping method, Evaluated on the Wikipedia test ground truth with hard cases.

<table>
<thead>
<tr>
<th>Type</th>
<th>Error rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.16</td>
<td>0.88</td>
<td>0.93</td>
<td>0.91</td>
<td>80.12%</td>
</tr>
<tr>
<td>LSTM(9)</td>
<td>0.19</td>
<td>0.82</td>
<td>0.97</td>
<td>0.89</td>
<td>80.00%</td>
</tr>
<tr>
<td><strong>Bi-LSTM(15)</strong></td>
<td><strong>0.07</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.97</strong></td>
<td><strong>0.96</strong></td>
<td><strong>91.39%</strong></td>
</tr>
<tr>
<td>NLTK default</td>
<td>0.15</td>
<td>0.87</td>
<td>0.98</td>
<td>0.92</td>
<td>84.87%</td>
</tr>
<tr>
<td>NLTK trained, customized</td>
<td>0.09</td>
<td>0.94</td>
<td>0.97</td>
<td>0.95</td>
<td>89.07%</td>
</tr>
<tr>
<td>spaCy parser default</td>
<td>0.23</td>
<td>0.81</td>
<td>0.94</td>
<td>0.87</td>
<td>73.53%</td>
</tr>
<tr>
<td>spaCy senter default</td>
<td>0.16</td>
<td>0.90</td>
<td>0.92</td>
<td>0.91</td>
<td>80.84%</td>
</tr>
<tr>
<td>spaCy senter trained</td>
<td>0.18</td>
<td>0.88</td>
<td>0.92</td>
<td>0.90</td>
<td>79.44%</td>
</tr>
</tbody>
</table>

- Bi-LSTM has the best precision
- LSTM: low precision due to false positive predictions
- trained NLTK second best
- spaCy parser - dependency tree is broken by abbreviations, quotes
Test on hard cases and random sentences from the Wikipedia corpus

- All systems perform better on random sentences
- LSTM improved most
⇒ SBD highly syntax dependent

<table>
<thead>
<tr>
<th>Context window</th>
<th>error rate</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test set with hard cases (2928 sentences)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bi-LSTM(17)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for each basic unit</td>
<td>0.0737</td>
<td>0.9534</td>
<td>0.9703</td>
<td>0.9617</td>
<td>90.95%</td>
</tr>
<tr>
<td>for punctuations</td>
<td>0.0880</td>
<td>0.9349</td>
<td>0.9738</td>
<td>0.9540</td>
<td>90.37%</td>
</tr>
<tr>
<td>for sentence end marks</td>
<td><strong>0.0699</strong></td>
<td><strong>0.9537</strong></td>
<td><strong>0.9741</strong></td>
<td><strong>0.9638</strong></td>
<td><strong>91.36%</strong></td>
</tr>
<tr>
<td>LSTM(9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for sentence end marks</td>
<td>0.1707</td>
<td>0.8497</td>
<td>0.9719</td>
<td>0.9067</td>
<td>81.90%</td>
</tr>
<tr>
<td>Test set with random cases (3267 sentences)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bi-LSTM(17)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for each basic unit</td>
<td>0.0182</td>
<td>0.9867</td>
<td>0.9950</td>
<td>0.9908</td>
<td>98.03%</td>
</tr>
<tr>
<td>for punctuations</td>
<td>0.0222</td>
<td>0.9830</td>
<td>0.9947</td>
<td>0.9888</td>
<td>97.68%</td>
</tr>
<tr>
<td>for sentence end marks</td>
<td><strong>0.0145</strong></td>
<td><strong>0.9882</strong></td>
<td><strong>0.9972</strong></td>
<td><strong>0.9927</strong></td>
<td><strong>98.28%</strong></td>
</tr>
<tr>
<td>LSTM(9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for sentence end marks</td>
<td>0.0511</td>
<td>0.9561</td>
<td>0.9878</td>
<td>0.9717</td>
<td>93.88%</td>
</tr>
</tbody>
</table>

The bi-LSTM models are trained on 500,000 Wikipedia sentences for 10 epochs the LSTM model on 3,787,371 sentences for 10 epochs.
With different methods for creating the context window we optimize the memory usage and the training time.

<table>
<thead>
<tr>
<th>Training data size/2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>token based</th>
<th></th>
<th>character based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>full text</td>
<td>punct.</td>
<td>end marks</td>
<td>full text</td>
</tr>
<tr>
<td>LSTM(9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7,000</td>
<td>75.41</td>
<td>75.52</td>
<td>75.48</td>
<td>78.59</td>
</tr>
<tr>
<td>500,000</td>
<td>80.02</td>
<td>78.28</td>
<td>79.30</td>
<td>81.11</td>
</tr>
<tr>
<td>3,780,371</td>
<td>80.28</td>
<td>82.67</td>
<td>79.99</td>
<td>81.40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>bi-LSTM(17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7,000</td>
<td>88.59</td>
<td>87.29</td>
<td>87.47</td>
<td>85.47</td>
</tr>
<tr>
<td>500,000</td>
<td>89.83</td>
<td>90.45</td>
<td>91.25</td>
<td>91.90</td>
</tr>
<tr>
<td>3,780,371</td>
<td>89.90</td>
<td>91.32</td>
<td>91.54</td>
<td>91.54</td>
</tr>
</tbody>
</table>

**Graph on the slide 22**

<table>
<thead>
<tr>
<th>Number of sentences</th>
<th>Context window created for every</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>character</td>
</tr>
<tr>
<td></td>
<td>Memory</td>
</tr>
<tr>
<td>7,000</td>
<td>28 MB</td>
</tr>
<tr>
<td>500,000</td>
<td>1.25 GB</td>
</tr>
<tr>
<td>3,780,371</td>
<td>9 GB</td>
</tr>
</tbody>
</table>
Rule-based approach

Find sentence boundaries with predefined rules:

*Sentence end mark + whitespace + capital letter:*

The sun is shining. The weather is sunny.

But: Mr. Brown is reading.

- Handcrafted rules, time and effort consuming
- Use a list of abbreviations
- Difficult to cover each case
- System is fragile due to too many special cases
- Hard to maintain and customize
Statistical approach

Collect statistics about:
- occurrences of tokens, punctuations,
- token length,
- casing
- collocations of tokens

Calculate probabilities and test some assumptions:
- frequent sentence starter
- token’s collocation with a period

Trainable with unsupervised training (NLTK)
- No annotation needed
Dependency tree

how the words within a sentence are related to each other

This is a sentence.

DET VERB DET DET NOUN
**Embeddings**

<table>
<thead>
<tr>
<th>Input</th>
<th>Word to index</th>
<th>Embeddings</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>7</td>
<td>[0.342 -0.025 -1.690 0.717]</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[-0.643 2.726 0.074 0.696]</td>
</tr>
<tr>
<td>the</td>
<td>4</td>
<td>[1.497 1.344 -0.965 3.453]</td>
</tr>
<tr>
<td>U</td>
<td>52</td>
<td>[-0.643 2.726 0.074 0.696]</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[0.342 -0.025 -1.690 0.717]</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>[0.222 -3.025 -1.650 0.237]</td>
</tr>
<tr>
<td>S</td>
<td>34</td>
<td>[0.543 -0.021 -1.950 0.377]</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>[0.222 -3.025 -1.650 0.237]</td>
</tr>
<tr>
<td>A</td>
<td>8</td>
<td>[-0.112 -0.032 1.690 0.754 ]</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>[0.222 -3.025 -1.650 0.237]</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[-0.643 2.726 0.074 0.696]</td>
</tr>
</tbody>
</table>

- Vector representation for words or characters
- Capture meaningful information
- Learned during the training process
A cat has nine lives. A journey of thousand miles begins with a single step.

Input: ['A', 'cat', 'has', 'nine', 'lives', '.']

Output: ['A', 'cat', 'has', 'nine', 'lives', '.']

<table>
<thead>
<tr>
<th>unit</th>
<th># occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>' '</td>
<td>3095</td>
</tr>
<tr>
<td>', '</td>
<td>295</td>
</tr>
<tr>
<td>'a'</td>
<td>1563</td>
</tr>
<tr>
<td>'of'</td>
<td>512</td>
</tr>
<tr>
<td>'begins'</td>
<td>2</td>
</tr>
<tr>
<td>'lives'</td>
<td>63</td>
</tr>
<tr>
<td>'......'</td>
<td></td>
</tr>
<tr>
<td>'miles'</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>index</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PAD</td>
<td>0</td>
</tr>
<tr>
<td>_UNK</td>
<td>1</td>
</tr>
<tr>
<td>_DIG</td>
<td>2</td>
</tr>
<tr>
<td>' '</td>
<td>3</td>
</tr>
<tr>
<td>', '</td>
<td>4</td>
</tr>
<tr>
<td>'a'</td>
<td>5</td>
</tr>
<tr>
<td>'of'</td>
<td>6</td>
</tr>
<tr>
<td>'......'</td>
<td></td>
</tr>
</tbody>
</table>

Input: [[5, 3, 9, 3, 13, 3, 16, 3, 23, 7], [5, 3, 1, 3, 6, 3, 11, 3, 1, 14, 3, 6, 3, 5, 3, 12, 3, 19, 7]]

Output: [[0, 0, 0, 5], [0, 0, 5, 3], [0, 5, 3, 9], [5, 3, 9, 3, 13], [3, 9, 3, 13, 3], [9, 3, 13, 3, 16], [3, 13, 3, 16, 3], [13, 3, 16, 3, 23], [3, 16, 3, 23, 7]]

Process of data preparation
F1-score

\[
\text{F1-score} = \left( \frac{\text{Precision}^{-1} + \text{Recall}^{-1}}{2} \right)^{-1} = \frac{2}{\frac{(FP+TP) + (FN+TP)}{TP}} = \frac{2TP}{FP+TP + FN+TP}
\]

\[
= \frac{2TP^2}{TP(FP+TP) + TP(FN+TP)}
\]

\[
= \frac{2TP^2}{(FP+TP)(FN+TP)}
\]

\[
= \frac{TP}{FP+TP} \cdot \frac{TP}{FN+TP}
\]

\[
= 2 \frac{\text{Precision}}{\text{Recall}} \cdot \frac{\text{Recall}}{\text{Precision} + \text{Recall}}
\]
\[ F_{\beta} \text{-score} = \frac{\text{Precision} \cdot \text{Recall}}{(1 + \beta^2) \cdot \beta^2 \cdot \text{Precision} + \text{Recall}} \]

apply additional weight \( \beta \) to consider recall \( \beta \)-times more than precision

\( \beta > 1 \) weighs recall higher than precision
Threshold

Threshold $\in [0.01, 1[$
steps of 0.01
Calculate F1-score
Save the threshold belonging to the best F1-score
Training setup

- batch size of 256: number of context windows in one iteration of the training,
- embeddings size of 256: the length of the feature vector encoding a basic unit
- number of units 128: the LSTM layer includes this number of neurons (dimension of the hidden layer)
- number of layers 1
- learning rate 0.01: the size of the steps in adjusting the weights during the training in order to minimize loss
- stochastic gradient descent (SGD) optimizer algorithm to minimize loss
- cross entropy loss function to measure the correctness of the prediction during the training
Early stopping

Loss
- difference between the labels and the model output

Validation
- Calculate loss after a certain number of epochs

Early stopping
- If the validation loss is not decreasing, stop the training process

Patience
- Number of epochs we wait before we stop the training
The average temperature is 20.7 °C. Most rain falls in the winter.

---

<table>
<thead>
<tr>
<th>For each basic unit</th>
<th>LSTM</th>
<th>bi-LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
</tr>
<tr>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
</tr>
<tr>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
</tr>
<tr>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
</tr>
<tr>
<td>The</td>
<td>The</td>
<td>The</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For punctuations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>20</td>
<td>.</td>
</tr>
<tr>
<td>20</td>
<td>.</td>
<td>7</td>
</tr>
<tr>
<td>°</td>
<td></td>
<td>c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For sentence end marks</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>20</td>
<td>.</td>
</tr>
<tr>
<td>20</td>
<td>.</td>
<td>°</td>
</tr>
<tr>
<td>°</td>
<td>c</td>
<td>.</td>
</tr>
</tbody>
</table>

Most rain falls in the winter.
Character based context windows

For each basic unit

<table>
<thead>
<tr>
<th>PAD</th>
<th>PAD</th>
<th>PAD</th>
<th>PAD</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>T</td>
<td>h</td>
</tr>
</tbody>
</table>

For punctuations

<table>
<thead>
<tr>
<th>2</th>
<th>0</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>.</td>
</tr>
<tr>
<td>s</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>°</td>
<td>c</td>
</tr>
</tbody>
</table>

For sentence end marks

<table>
<thead>
<tr>
<th>2</th>
<th>0</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>°</td>
<td>c</td>
</tr>
<tr>
<td>°</td>
<td>c</td>
<td>.</td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance in English

Punkt from NLTK
- Error rate 1.02% on Brown and 1.65% on WSJ

spaCy parser
- F1-score 0.90 on OntoNotes 5.0

Our baseline

<table>
<thead>
<tr>
<th></th>
<th>Error rate</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>21%</td>
<td>0.88</td>
</tr>
<tr>
<td>Wall Street Journal</td>
<td>24%</td>
<td>0.86</td>
</tr>
</tbody>
</table>

large corpus comprising various genres of text (news, conversational telephone speech, weblogs, usenet newsgroups, broadcast, talk shows)
Baseline algorithm evaluated on different corpora

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Error rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>arXiv</td>
<td>0.08</td>
<td>0.94</td>
<td>0.97</td>
<td>0.96</td>
<td>92.30%</td>
</tr>
<tr>
<td>Europarl</td>
<td>0.07</td>
<td>0.97</td>
<td>0.95</td>
<td>0.96</td>
<td>90.98%</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>0.16</td>
<td>0.88</td>
<td>0.93</td>
<td>0.91</td>
<td>80.12%</td>
</tr>
<tr>
<td>Brown</td>
<td>0.21</td>
<td>0.94</td>
<td>0.83</td>
<td>0.88</td>
<td>67.43%</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.24</td>
<td>0.83</td>
<td>0.89</td>
<td>0.86</td>
<td>69.78%</td>
</tr>
</tbody>
</table>

- Strong lower bound on Europarl and arXiv
- Formulas and variables at sentence end in arXiv do not cause errors
- Abbreviations like Senator J. W. Fulbright lower precision
- Titles without any punctuation cause missed sentence boundaries
- Highest error rate for Brown and WSJ
Context window types

- the token- and character-based approaches for creating the input
- three different context window method
  - the character-based models overall better
  - context window for each unit inefficient for memory and time
  - context window for every punctuation mark reduces training data
Why we need sentences?

Used in other tasks within NLP

- POS-tagging

<table>
<thead>
<tr>
<th>Book</th>
<th>verb</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>determiners</td>
</tr>
<tr>
<td>room.</td>
<td>noun</td>
</tr>
</tbody>
</table>

- Text correction

<table>
<thead>
<tr>
<th>How much is the temperature?</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the temperature?</td>
</tr>
</tbody>
</table>

- Sentiment analysis:

<table>
<thead>
<tr>
<th>It’s great!</th>
<th>😄</th>
</tr>
</thead>
<tbody>
<tr>
<td>We will be there in 5 Min.</td>
<td>😞</td>
</tr>
<tr>
<td>It was a bad game.</td>
<td>😞</td>
</tr>
</tbody>
</table>

- Machine translation

<table>
<thead>
<tr>
<th>Pink ist meine Lieblingsfarbe.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pink is my favourite color.</td>
</tr>
</tbody>
</table>
Gates in LSTM

Forget gate:
- $f_f = \sigma(W_f(h_t, x) + b_f)$
- filters the old state $c_t$ and decides what information to discard: $c_t \odot f_f$

Input gate:
- $f_i = \sigma(W_i(h_t, x) + b_i)$
- decides what information to let through from the current input $x$

New candidate:
- $f_{nc} = \tanh(W_{nc}(h_t, x) + b_{nc})$
- Regulate the input between -1 and +1 with $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
Gates in LSTM

Output gate:
- $f_o = \sigma(W_o(h_t, x) + b_o)$
- calculates how much of the input is used in our output $h_{t+1}$

Cell state:
- $c_{t+1} = (c_t \otimes f_f) \oplus (f_i \otimes f_{nc})$
- a kind of memory chaining through all the time steps.

Hidden state:
- $h_{t+1} = \tanh(c_{t+1}) \otimes f_o$
- Output of the network
Dimensionality and LSTM computation

- $f_f = \sigma(W_f(h_t, x) + b_f)$
- $f_i = \sigma(W_i(h_t, x) + b_i)$
- $f_{nc} = \tanh(W_{nc}(h_t, x) + b_{nc})$
- $f_o = \sigma(W_o(h_t, x) + b_o)$
- $c_{t+1} = (c_t \otimes f_f) \oplus (f_i \otimes f_{nc})$
- $h_{t+1} = \tanh(c_{t+1}) \otimes f_o$
LSTM Model architecture

Raw text → Input layer → Embedding layer → LSTM 128 neurons → Hidden layer → Linear layer → Softmax layer → Output layer

- **Input layer**
- **Embedding layer**
- **Hidden layer**
- **Linear layer**
- **Softmax layer**
- **Output layer**

- **y = xA + b**
- **softmax(v̂)**

- **Context windows**
- **Feature vectors**
- **Hidden state**
- **Logits**
- **Probs**
- **Labels**

- **batch size**
- **emb. dim**
- **hidden. dim**
- **x**
- **context size**
- **label size**