# Automatic Correction of Misaligned Spaces and Typos Using Deep Learning

## Master Thesis

Mostafa M. Mohamed

Albert-Ludwig Universtät Freiburg

July 19, 2018

# Problem statement

Design a model that automatically classifies and corrects tokenization errors, which are one of four types:

1. Missing spaces (like ‘HelloWorld’)
2. Adding wrong spaces (like ‘Hello Wor ld’)
3. Line break hyphenation (like ‘Hello Wo-[newline]rld’)
4. Garbled characters (like ‘Hello Warld’)

# Motivation

- A variety of systems process text corpora as individual words.
  - Text search
- A variety of sources don't store text in a fully recoverable format
  - PDF
  - Images
- Corpora extracted with such errors will be harder to deal with, hence the urge to automatically fix them in advance.

# Outline

# Outline

# Edit Distance

- Edit distance between $T$ and $S$ is the number of operations to be changed from $T$ to transform it into $S$.
- Characters are changed by addition or deletion operations.
- Edit distance can be traced to find edit operations from $T$ to $S$.
- $\mathit{EditOperations}(\text{‘abcx’}, \text{‘acxy’}) = \{(\text{DEL}, 2, \text{‘b’}), (\text{ADD}, 5, \text{‘y’})\}$

# Outline

# Baseline approaches

1. Greedy approach: Keep matching as much words as possible from beginning of text, according to a dictionary.
2. 3-Gram Markov Model: Similar to the main (bicontext) approach; replaces the language model by a simpler probabilistic model.

# Greedy approach

Correcting ‘The reis someonegoing.’

- Use a stream of the non-delimiter characters.
- The stream is ‘Thereissomeonegoing.’
- Extract the words ‘There’, ‘is’, ‘someone’, ‘going’, ‘.’
- Join them into ‘There is someone going.’

# Drawbacks with greedy approach

Correcting ‘The remainingfood isdelicious.’

- The stream is ‘Theremainingfoodisdelicious.’
- Extract the words ‘There’, ‘main’, ‘in’, ‘g’, ‘food’, ‘is’, ‘delicious’, ‘.’
- Join them into ‘There main in g food is delicious.’

Which is a wrong correction, because matching ‘There’ greedily made it hard to fix the remaining sentence.

# Outline

# Proposed dynamic programming approach

The components of the dynamic programming approach:

- Grouping consequent tokens
- Retokenizing each group separately
- Scoring tokens

The approach matches globally as many words as possible, according to a dictionary.

# Example

’Thisis the mostbasic ex amp le f or the algorithm.”

would idealy be grouped as:

’(Thisis) (the) (mostbasic) (ex amp le) (f or) (the) (algorithm)(.)”

and then the characters are retokenized as:

’(This is) (the) (most basic) (example) (for) (the) (algorithm)(.)”

and finally, the acquired fixed text:

’This is the most basic example for the algorithm.”

# Drawbacks with dynamic programming approach

Correcting:

‘The rest are at theage oftwenty.’

Can be fixed into:

‘There stare at the age of twenty.’

A wrong correction, as the chosen English words don’t fit in context.

# Outline

# Deep learning background

1. Language models.
2. Recurrent neural networks.
3. Text generation.
4. Beam search.

# Language model

A language model $M$ is an estimator of how likely a string $s$ to occur as a string of a specific language.

- Forward prediction: $p_f(y|X)$ probability of the character $y$ to come after $X$, by feeding the last $h$ characters from $X$ as input.
- Backward prediction: $p_b(y|X)$ probability of the character $y$ to come before $X$, by feeding the first $h$ characters from $X$ as input.

# Neural networks

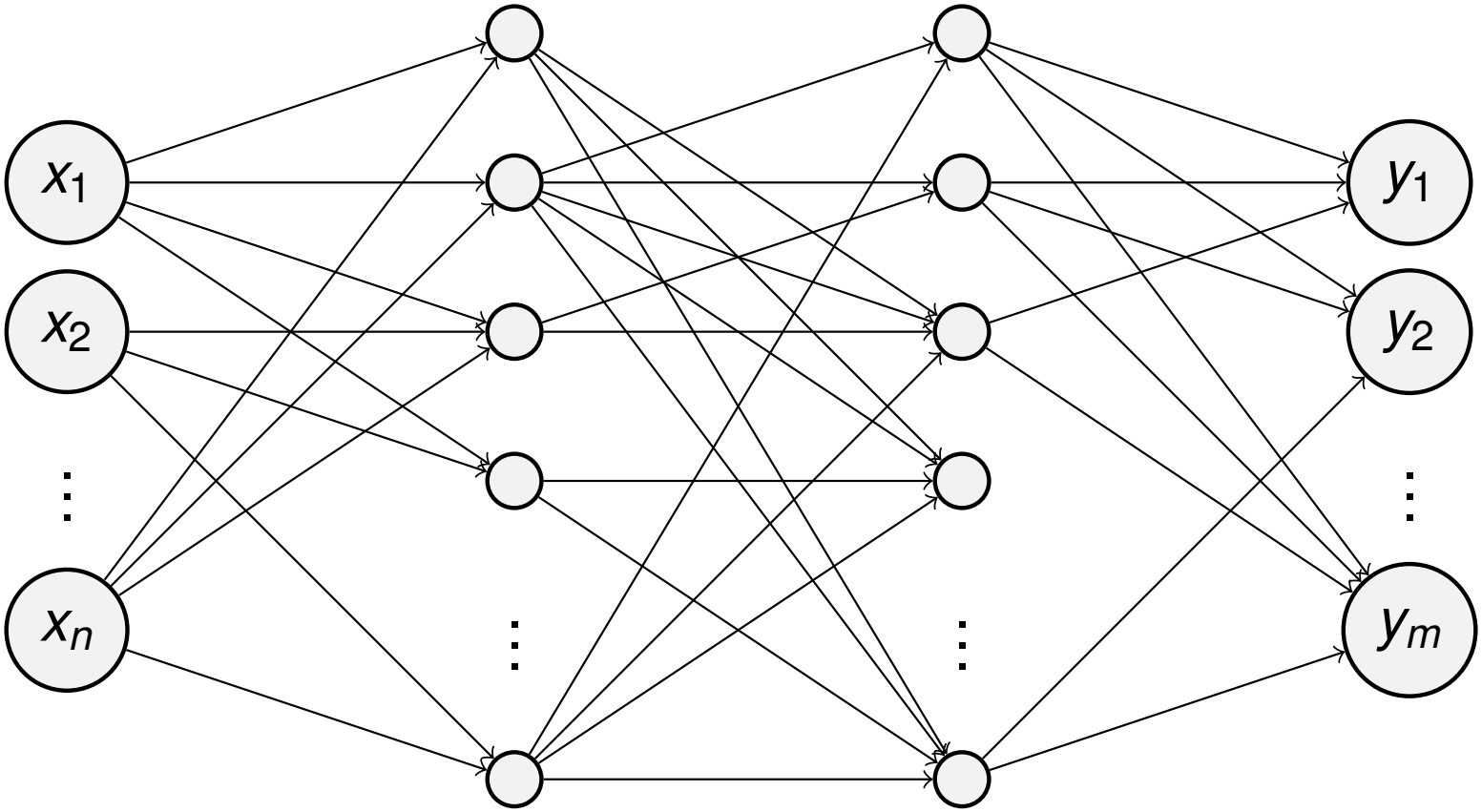


Figure : Neural network example, $x_1, \cdots, x_n$ are the input neurons, $y_1, \cdots, y_m$ are the output neurons. The remaining neurons are the hidden ones.

# RNN language model

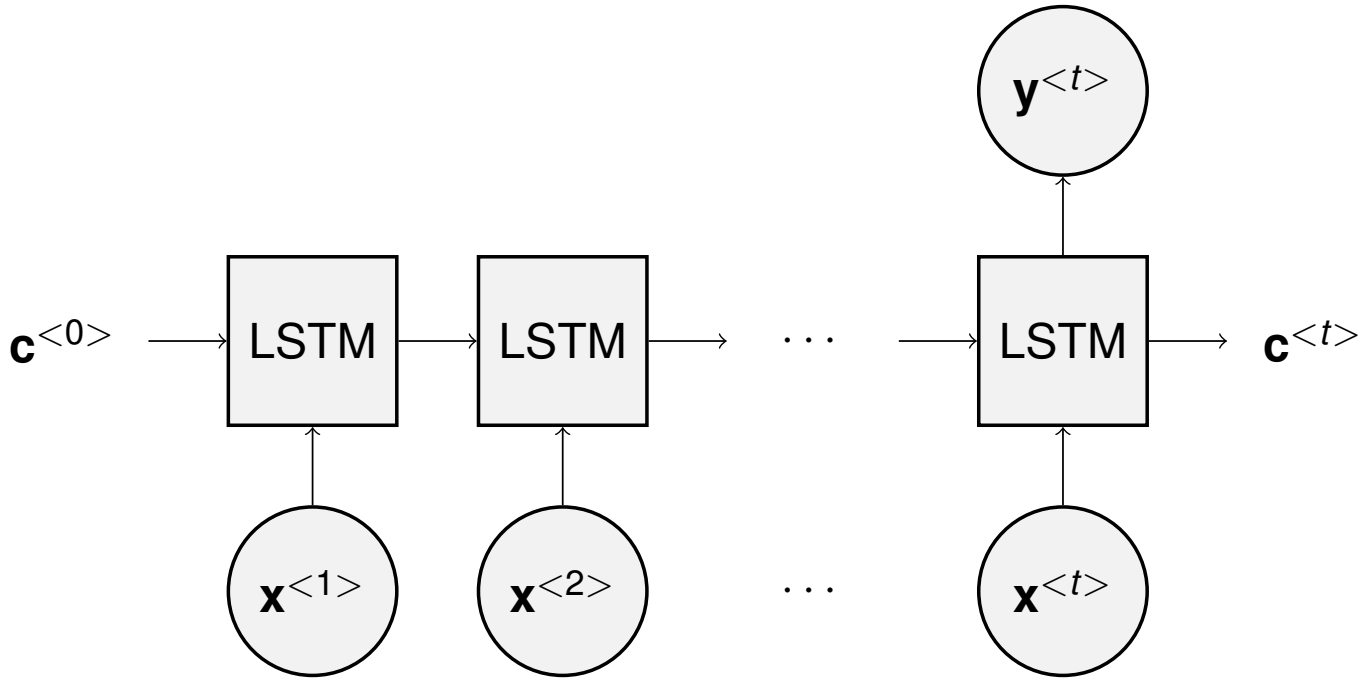


Figure : Many-to-one LSTM recurrent neural network, $\mathbf{x}^{<t>}$ is the $t$-th element in the input sequence, $\mathbf{c}^{<t>}$ is the $t$-th activation value. The output of the network is $\mathbf{y}^{<t>}$.

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'
- 'It is found that th' generates 'e'

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'
- 'It is found that th' generates 'e'
- 'It is found that the' generates ' ' (space)

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'
- 'It is found that th' generates 'e'
- 'It is found that the' generates ' ' (space)
- 'It is found that the ' generates 'f'

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'
- 'It is found that th' generates 'e'
- 'It is found that the' generates ' ' (space)
- 'It is found that the ' generates 'f'
- 'It is found that the f' generates 'i'

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'
- 'It is found that th' generates 'e'
- 'It is found that the' generates ' ' (space)
- 'It is found that the ' generates 'f'
- 'It is found that the f' generates 'i'
- 'It is found that the fi' generates 'r'

# Text generation with language models

Language models can generate texts, by continuously applying the function $S \leftarrow S \circ \mathit{WeightedSample}_x p_f(x|S)$. For example:

- 'It is found that ' generates 't'
- 'It is found that t' generates 'h'
- 'It is found that th' generates 'e'
- 'It is found that the' generates ' ' (space)
- 'It is found that the ' generates 'f'
- 'It is found that the f' generates 'i'
- 'It is found that the fi' generates 'r'
- ...

# Generated text

‘It is found that the first album in the canton of the top of the part of the main part of the most common and songs of the album and an anadring and particles and in the process of the city of the most of the planets in the south of the season of the main movie and ended in 1991. It is a species of started and the constitution of the name in the contract of the police match of the principal that is a common of the Army in the United States. It is also be a member of the Earth was an area of the people of the political canton of the state of the book in the and of the area of the simple of the song of the only became a hard and an area of the area of the song and an ended of the books.’

# Beam search

Beam search finds semi-optimal states in a gigantic state space, it $\cdots$

- works like breadth-first-search, keeping top $b$ states at each level.
- can reach more globally optimal solutions than greedy search.
- $b$ is referred to as the beam size.
- $b = 1$ will make beam search a greedy search.
- $b \to \infty$ will make beam search a full search.

# Outline

# Deep learning approaches

DL approaches utilize RNN language models to solve the problem.

1. States of two contexts
2. Bicontext approach
   - Occurrence function
   - Fixing decisions scores
   - Tuner
3. End-to-end approach

# States of two contexts

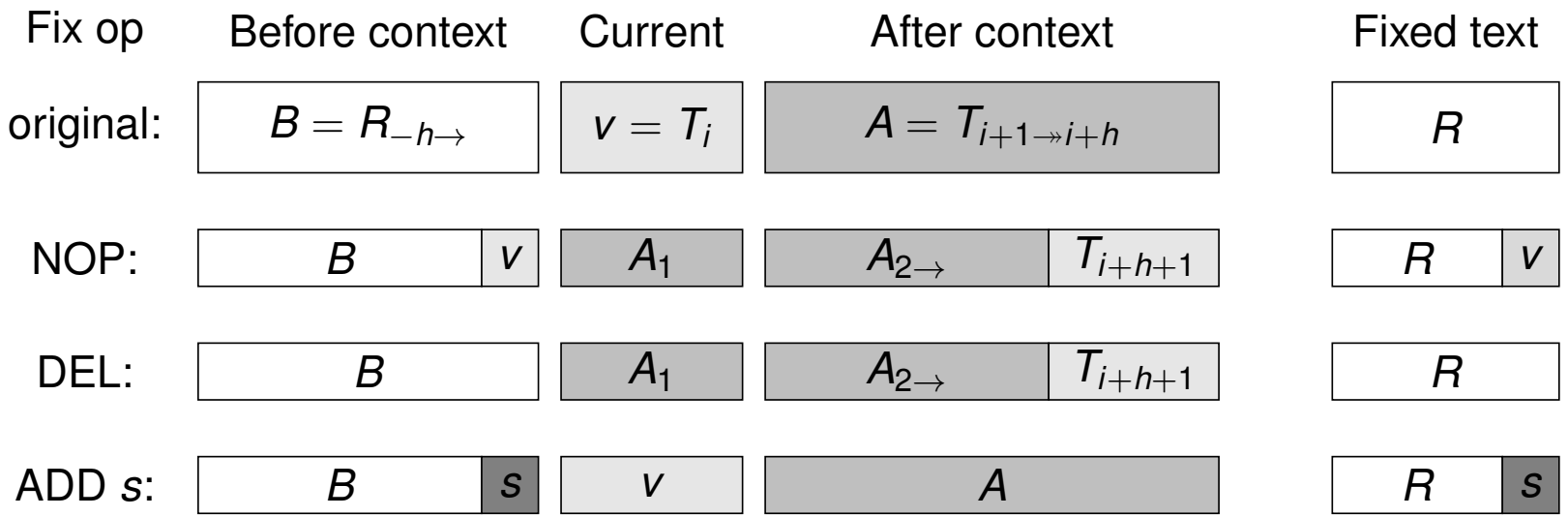


Figure : The effects of all fixing operations NOP, DEL and ADD, on the updates of states.

By encapsulating $(B, v, A, R, i)$ into a state $S$ (interpreted as corrections of prefixes), finding the most probable fixed string is a search problem.

# Occurrence function

Expectancy occurrence function, how likely is the context $X$ follows $Y$.

$$P_o(X, Y) := \frac{b(X, Y, s)}{s} \cdot \frac{f(X, Y, s)}{s} \tag{1}$$

Where,

$$\begin{aligned} b(X, Y, s) &:= \sum_{i=1}^{\min\{|X|+1, s\}} p_b(X_{-1}, \cdots, X_{-i}|Y) \\ f(X, Y, s) &:= \sum_{i=1}^{\min\{|Y|+1, s\}} p_f(Y_1, \cdots, Y_i|X) \end{aligned} \tag{2}$$

Expected number of $b(X, Y, s)$ characters from $X$ to come before $Y$.
Expected number of $f(X, Y, s)$ characters from $Y$ to come after $X$.
$p_f$ and $p_b$ are computed with the RNN language models.

# Text

‘The 1976 Summer$^{2}$ Paralympics$^{1}$ took$^{2}$ place$^{2}$ in Toronto, Ontario, Canada.
1,657 athletes from$^{2}$ 38$^{1}$ were$^{1}$ at$^{4}$ the$^{2}$ Games. People$^{2}$ with these$^{1}$ types of$^{4}$ disabilities$^{1}$ competed at the games$^{2}$: spinal$^{2}$ injury$^{2}$, amputee, blindness, and$^{1}$ Les Autres.’

---

‘The 1976 Summe r Paraly mpicstook pla ce in Toronto, Ontario, Canada.
1,657 athletes fro m 3awereat th e Games. Pe ople with thesetypes f̂ disabilitiescompeted at the gam es: sp inal i njury, amputee, blindness, andLes Autres.’

# Fixed text

‘The 1976 Summer Paralympics took place in Toronto, Ontario, Canada.
1,657 athletes from ~~38~~ were at the Games. People with these types of disabilities competed at the games: spinal injury, amputee, blindness, and Les Autres.’

# Example

- Pe ople with thesetypes $\hat{\text{f}}$ disabilitiescompete d at the gam es
- Peop le with these types of disabilities competed at the games

$B =$‘le with these types ’, $v =$‘f’, $A =$‘ disabilitiescompete’, $s =$‘o’

| op | $X$ | $Y$ | $f(X, Y, 3)$ | $b(X, Y, 3)$ | $3^2 \cdot P_o(X, Y)$ | $\log 9P_o$ |
|---|---|---|---|---|---|---|
| DEL | $B$ | $A$ | $10^{-6}$ | $6 \cdot 10^{-6}$ | $10^{-10}$ | $-23.68$ |
| NOP | $Bv$ | $A$ | 0.004 | 0.028 | $10^{-4}$ | $-9.09$ |
| NOP | $B$ | $vA$ | 0.014 | 0.0033 | $4 \cdot 10^{-6}$ | $-10.00$ |
| ADD | $Bs$ | $vA$ | 1.98 | 2.06 | 4.08 | **1.41** |
| ADD | $B$ | $svA$ | 2.59 | 1.33 | 3.44 | 1.24 |

# Decisions probabilities

Fixing edit operations are decided with scores:

$$\begin{aligned}
\log \mathbb{P}_{\text{DEL}} &= \log P_o(B, A) \\
\log \mathbb{P}_{\text{NOP}} &= \frac{1}{2} \log P_o(Bv, A) + \frac{1}{2} \log P_o(B, vA) \\
\log \mathbb{P}_{\text{ADD s}} &= \frac{1}{2} \log P_o(Bs, vA) + \frac{1}{2} \log P_o(B, svA)
\end{aligned} \tag{3}$$

# Decisions probabilities

With more tuning:

$$
\begin{aligned}
\log \mathbb{P}_{\text{DEL}} &= \log P_o(B, A) \\
\log \mathbb{P}_{\text{NOP},s\notin\Gamma} &= \frac{1}{2}\log P_o(Bv, A) + \frac{1}{2}\log P_o(B, vA) + 2\log 0.005 \\
\log \mathbb{P}_{\text{NOP},s\in\Gamma} &= \frac{1}{2}\log P_o(Bv, A) + \frac{1}{2}\log P_o(B, vA) \\
\log \mathbb{P}_{\text{ADD s},s\notin\Gamma} &= \frac{1}{2}\log P_o(Bs, vA) + \frac{1}{2}\log P_o(B, svA) + 2\log 0.005 \\
\log \mathbb{P}_{\text{ADD s},s\in\Gamma} &= \frac{1}{2}\log P_o(Bs, vA) + \log P_o(B, svA)
\end{aligned}
\tag{4}
$$

Where $\Gamma$ is the set of delimiters.

# Tuned decisions probabilities

The scores can be tuned with linear weights and biases:

$$
\begin{aligned}
\log \mathbb{P}_{\mathrm{DEL},v\in\Gamma} &= w_1 \log P_o(B, A) + b_1 \\
\log \mathbb{P}_{\mathrm{DEL},v\notin\Gamma} &= w_2 \log P_o(B, A) + b_2 \\
\log \mathbb{P}_{\mathrm{NOP},v\in\Gamma} &= w_3 \log P_o(Bv, A) + w_4 \log P_o(B, vA) + b_3 + b_4 \\
\log \mathbb{P}_{\mathrm{NOP},v\notin\Gamma} &= w_5 \log P_o(Bv, A) + w_6 \log P_o(B, vA) + b_5 + b_6 \\
\log \mathbb{P}_{\mathrm{ADD},s\in\Gamma} &= w_7 \log P_o(Bs, vA) + w_8 \log P_o(B, svA) + b_7 + b_8 \\
\log \mathbb{P}_{\mathrm{ADD},s\notin\Gamma} &= w_9 \log P_o(Bs, vA) + w_{10} \log P_o(B, svA) + b_9 + b_{10}
\end{aligned}
\tag{5}
$$

An optimizer is used to find optimal values for $\mathbf{w}, \mathbf{b}$.

# Outline

# Additional End-to-End Deep Learning Approach

In the bicontext approach:

- Language models are trained with correct texts.
- Tuner is trained with aligned pairs of correct and corrupt texts.
- Output is a list of scored fixing actions.

In the end-to-end approach:

- Input is aligned pairs of correct and corrupt texts.
- Classify/Score possible fixing actions.
- Everything else is learned by a neural network.

# Outline

# Evaluation Method

$\mathcal{C} := \mathit{EditOperations}(G, C)$ and $\mathcal{F} := \mathit{EditOperations}(G, F)$,
The fixer is a classifier of corruptions, where:

- Correctly identified corruptions, $TP = |\mathcal{C} \cap \overline{\mathcal{F}}| = |\mathcal{C} \setminus \mathcal{F}|$,
- Wrongly identified corruptions, $FP = |\mathcal{F} \cap \overline{\mathcal{C}}| = |\mathcal{F} \setminus \mathcal{C}|$,
- Undetected corruptions, $FN = |\mathcal{F} \cap \mathcal{C}|$

The classification of corruptions is evaluated using $F_1$-score.

$$\mathit{precision} = \frac{TP}{TP + FP}, \mathit{recall} = \frac{TP}{TP + FN}, F_1 = \frac{2PR}{P + R} \tag{6}$$

# Example

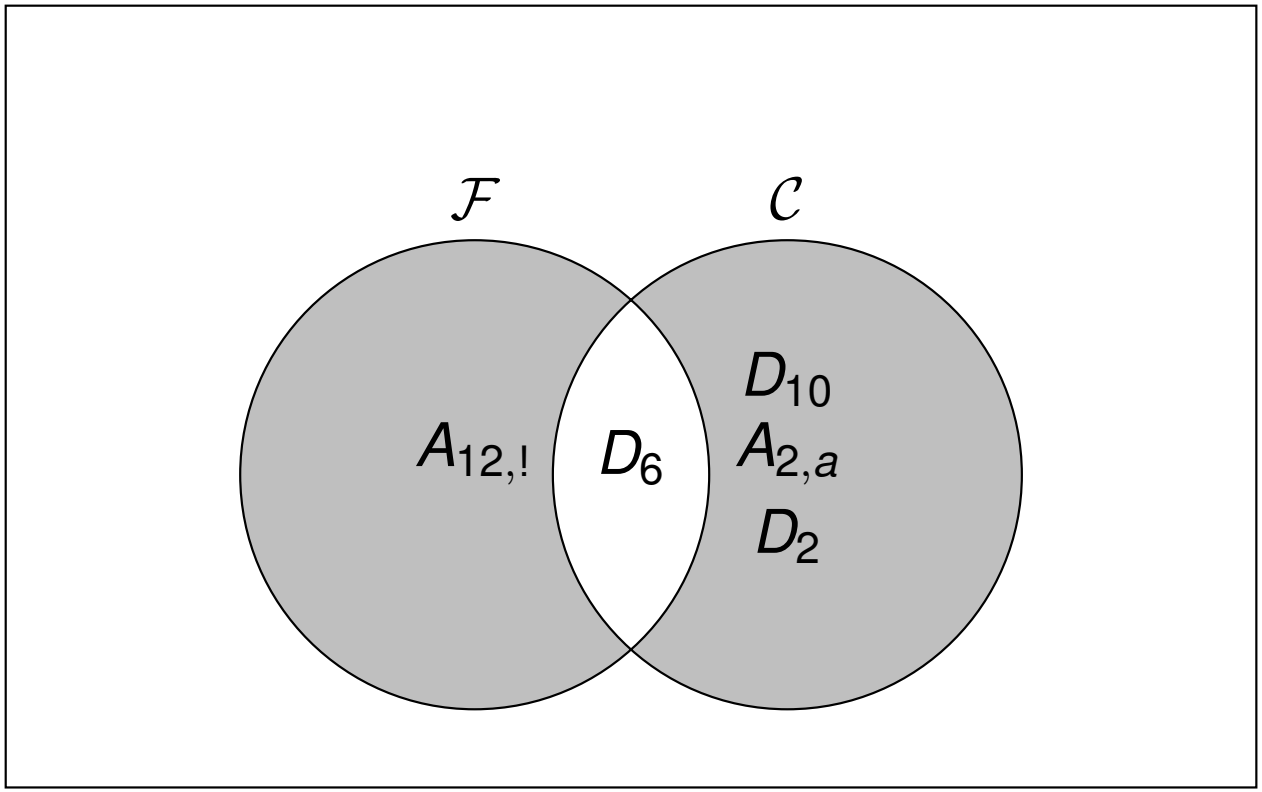


Figure : Simple Venn diagram of the two sets $\mathcal{F}$ and $\mathcal{C}$. Deletion and addition operations have the symbols D and A, respectively. $G =$ 'Hello world', $F =$ 'Helloworld!' and $C =$ 'Halloword'

# Datasets

- Two datasets were used in evaluation.
  - Reuters-21578
  - Simple-Wikipedia
- Corrupt texts were generated by an algorithm that corrupts nearly half the tokens, by a random tokenization mistake.

# Datasets

| Feature | Reuters-21578 | Simple-Wikipedia |
|---|---|---|
| # of articles | 19,043 | 131,566 |
| Avg. # of characters | 800 | 800 |
| Vocabulary | 49,847 | 348,924 |
| Vocabulary (freq $\geq 3$) | 24,986 | 130,191 |
| Domain | Economics | Various |

Table : Summary of the datasets

# Fixing Evaluations

| Type | Simple-Wikipedia | Reuters-21578 |
|---|---|---|
| Greedy baseline | 0.420 | 0.478 |
| 3-Gram Markov baseline | 0.468 | 0.331 |
| DP approach | 0.696 | 0.680 |
| End-to-End | 0.820 | 0.737 |
| Bicontext approach (128) | **0.886** | 0.854 |
| Bicontext approach (256) | - | **0.873** |

Table : Fixing evaluations on both datasets, measured by the mean $F_1$-scores. The result are 3-fold cross validated.

# Fixing Evaluations

| Type | Simple-Wikipedia | Reuters-21578 |
|---|---|---|
| Using default tuner weights | 0.819 | 0.814 |
| Beam size 1 | 0.872 | 0.846 |
| 50 Epochs | 0.876 | 0.840 |
| 100 Epochs, beam size 2 | 0.886 | 0.858 |
| Beam size 4 | 0.887 | 0.856 |
| More data ($\times 2$) | 0.903 | - |
| No typos | **0.963** | **0.919** |

Table : Experiments with changing one setting per experiment, using the bicontext model. The shown values are the 3-fold cross validated $F_1$-scores .

# Conclusions

- Two dictionary-based approaches and three learning-based approaches.
- Dynamic programming approach is the best dictionary-based.
- Deep learning-based bicontext model is the best model.
- Deep learning-based bicontext model maintains two contexts and search the best result.
- The results of detecting and fixing corruptions have $F_1$ score 90% up to 96%.

# Questions

Thanks for listening :) Any questions?

# Demo

# References


- Goodfellow, Ian and Bengio, Yoshua and Courville, Aaron and Bengio, Yoshua, *Deep Learning*, 2016
- Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford, *Introduction to Algorithms*, 2009
- Olson, David L. and Delen, Dursun, *Advanced Data Mining Techniques*, 2008:
- Norvig P., *Paradigms of Artificial Intelligence Programming: Case Studies in Common LISP*, 1992.
- Graves A., *Generating sequences with recurrent neural networks*, 2013.
- `https://dumps.wikimedia.org/simplewiki/`
- `https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection`

# Outline

# Trie data-structure

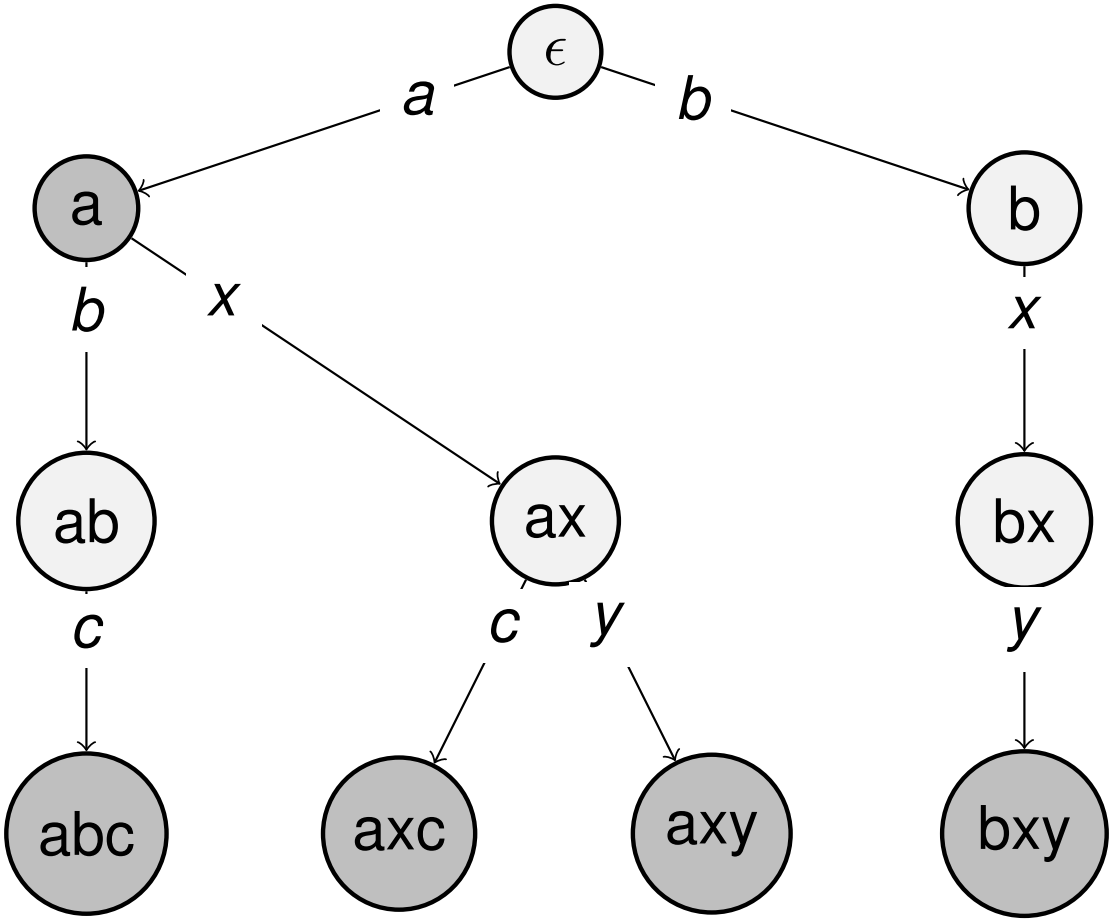


Figure : Trie datastructure sample, with five inserted strings: ‘a’, ‘abc’, ‘axc’, ‘axy’, ‘bxy’

# Outline

# Tokens scoring

All tokens will be scored according to this function:

$$\text{score}(w) = \begin{cases} \zeta & \text{if } w \text{ is a special string} \\ 0 & \text{if } e(w, D) > 1 \\ \varphi^{e(w,D)}(\alpha|\hat{w}|^2 + \beta|\hat{w}| + \gamma) & \text{otherwise} \end{cases} \tag{7}$$

Where,

- $\hat{w}$ is the nearest matching word in the given dictionary
- $e(w, D)$ is the edit distance to the nearest matching word
- $\alpha = 1.15, \beta = 0.1, \gamma = 1, \zeta = 2, \varphi = 0.5$ are hyperparameters

# Tokens scoring example

| Query $q$ | matched $w$ | length $\lvert w \rvert$ | $e(w, D)$ | Score($q$) | $\frac{Score(q)}{\lvert w \rvert}$ |
|---|---|---|---|---|---|
| Hello | Hello | 5 | 0 | 30.25 | 6.05 |
| world | world | 5 | 0 | 30.25 | 6.05 |
| **warld** | world | 5 | 1 | 15.125 | 3.025 |
| war | war | 3 | 0 | 11.65 | 3.88 |
| to | to | 2 | 0 | 5.8 | 2.9 |
| **td** | to | 2 | 1 | 2.9 | 1.45 |
| today | today | 5 | 0 | 30.25 | 6.05 |
| **tday** | today | 5 | 1 | 15.125 | 6.05 |
| the | the | 3 | 0 | 11.65 | 3.88 |
| query | - | 0 | $> 1$ | 0 | 0 |
| **ad** | - | 0 | $> 1$ | 0 | 0 |

Table : Tokens' scores, $q$ is a given query word, $w$ is the nearest match for it from the dictionary $D$. The dictionary contains the words 'Hello', 'world', 'war', 'to', 'today' and 'the'.

# Retokenization

Retokenization of tokens $T_1, \cdots, T_n$, which are joined into a big word $w$, is done by retokenizing any suffix of $w$ according to:

$$B_w[i] = \max_{i<j\leq|w|+1} \{B_w[j] \cdot \theta_{i,j} + (1 - \theta_{i,j}) \cdot Score(w_{i\rightarrow j})\} \tag{8}$$

Where $\theta_{i,j} = \frac{|w|-j+1}{|w|-i+1}$ is a normalizing term.

- $B_w[i]$ solves the suffix $w_{i\rightarrow}$
- Choose the first token ending at position $j$, using $Score(w_{i\rightarrow j})$
- Solve the remaining shorter suffix recursively, using $B_w[j]$

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d |
|---|---|---|---|---|---|---|---|---|---|---|
| *i* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| *B* | 22.7 | 12.8 | 10.2 | 11.6 | 12.8 | 15.1 | 1.5 | 0.0 | 0.0 | 0.0 |
| *nxt* | 6 | 6 | 4 | 6 | 6 | 11 | 9 | 9 | 10 | 11 |

Table : Values of retokenization of 'Hellowarld' into 'Hello world'. The grey cells mark the beginning of the new tokens according to *nxt*.

# Grouping

- Grouping $T_1, \cdots, T_n$ is done by grouping any suffix of tokens $T_i, \cdots, T_n$ according to:

$$F[i] := \max_{1 \leq d \leq \omega, n+1-i} \{F[i+d] + G(i,d)\} \tag{9}$$

Where $G$ retokenizes the group of chosen tokens $T_i, \cdots, T_{i+d-1}$:

$$G(i,d) = \max \begin{cases} \text{Score}(T_i \circ T_{i+1} \circ \cdots \circ T_{i+d-1}) & \text{if not 0} \\ \text{Retokenize}(T_i \circ T_{i+1} \circ \cdots \circ T_{i+d-1}) & \text{otherwise} \end{cases} \tag{10}$$

- The remaining suffix of tokens is solved recursively by $F[i+d]$.
- The group size is restricted up to $\omega \leq 8$ tokens.

# Grouping in action

| $i$ | $d$ | Tokens group | Retokenization | Joined | Score $G(i, d)$ |
|---|---|---|---|---|---|
| 1 | 1 | He | He | - | 0.0 |
| 1 | 2 | He, llowarl | Hello, war | - | 21.21 |
| **1** | **3** | He, llowarl, d | Hello, world | - | **22.69** |
| 2 | 1 | llowarl | l, to, war | - | 6.49 |
| **2** | **2** | llowarl, d | l, to, world | - | **10.19** |
| 2 | 3 | llowarl, d, td | l, to, world, to | - | 8.72 |
| 3 | 1 | d | d | - | 0.0 |
| 3 | 2 | d, td | d, to | - | 1.93 |
| **3** | **3** | d, td, ay | d, today | - | **7.92** |
| 4 | 1 | td | - | to | 2.90 |
| **4** | **2** | td, ay | - | today | **9.90** |
| **5** | **1** | ay | ay | - | **0.0** |

Table : $G(i, d)$ for the corrupt text: 'He llowarl d td ay'

# Grouping in action

| | | He | llowarl | d | td | ay |
|---|---|---|---|---|---|---|
| | $G$ | 1 | 2 | 3 | 4 | 5 |
| He | 1 | 0.0 | 21.21 | 22.69 | - | - |
| | 2 | - | 6.49 | 10.19 | 8.72 | - |
| | 3 | - | - | 0.0 | 1.93 | 7.92 |
| td | 4 | - | - | - | 2.90 | 9.90 |
| | 5 | - | - | - | - | 0.0 |

| | He | llowarl | d | td | ay |
|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 |
| $F$ | 32.6 | 20.1 | 9.9 | 9.9 | 0.0 |
| *nxts* | 4 | 4 | 4 | 6 | 6 |

Table : Functions $F[i]$ and $G(i, d)$ for the tokens. Grey cells mark the beginnings of the groups. The text is grouped into '(He llowarl d) (td ay)'

# Outline

# Backpropagation

- Weights on the connections can be adapted to adjust the output
- The network's ability to approximate a function is measured by a loss function
- The network trains to approximate a function by iteratively adapting the weights to minimize the loss function, using backpropagation
- For classification tasks, the loss used is *categorical cross-entropy*

$$\mathcal{J}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{m} \sum_{i=1}^{m} -y_i \log \hat{y}_i \tag{11}$$

# Outline

# 3-Gram Markov model

- The second baseline approach uses the bicontext approach
- It replaces the RNN language model by 3-Gram Markov model, where the probabilities are given by:

$$p(v|C) = \frac{\text{count}(C \circ v)}{\text{count}(C) + \epsilon} \tag{12}$$

- This could be enhanced by smoothing techniques.

# Outline

# Beam search example

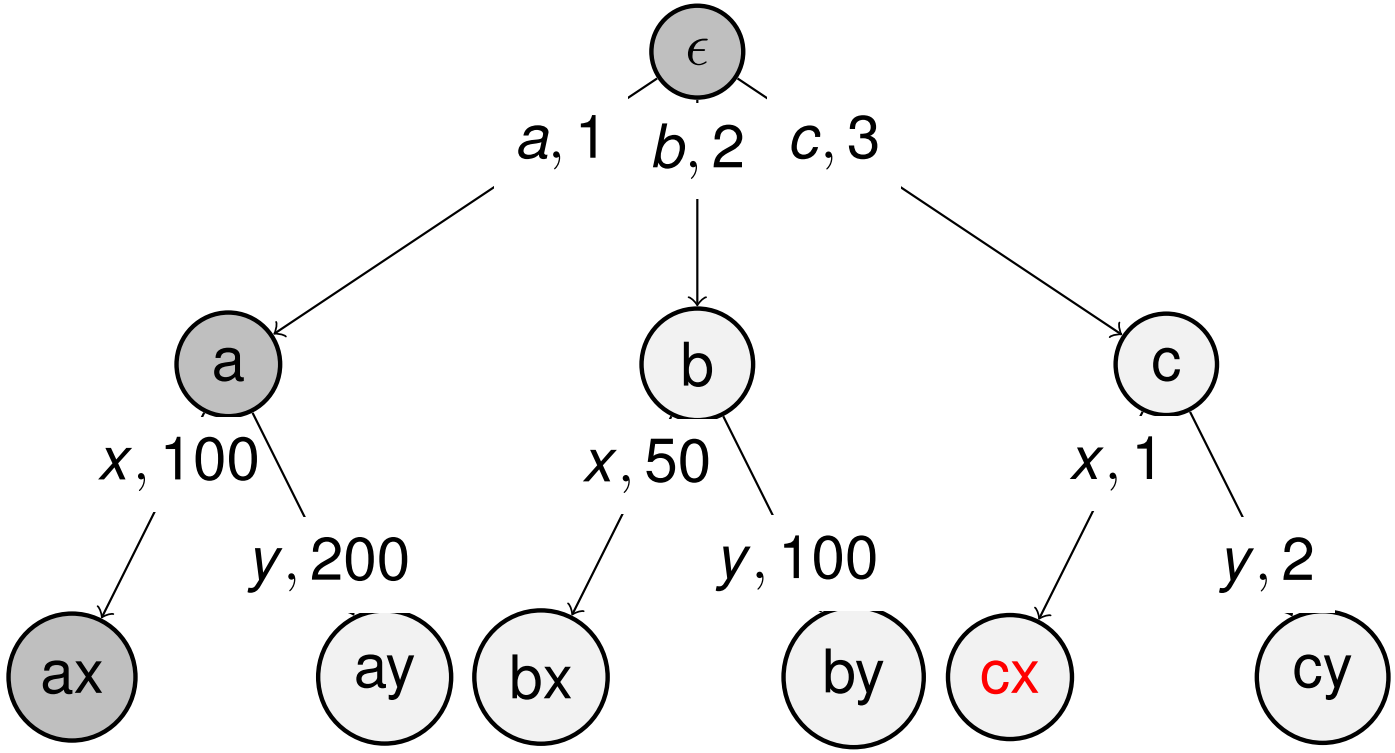


Figure : Beam search, beam size = 1. Traversed nodes are marked in grey.

# Beam search example

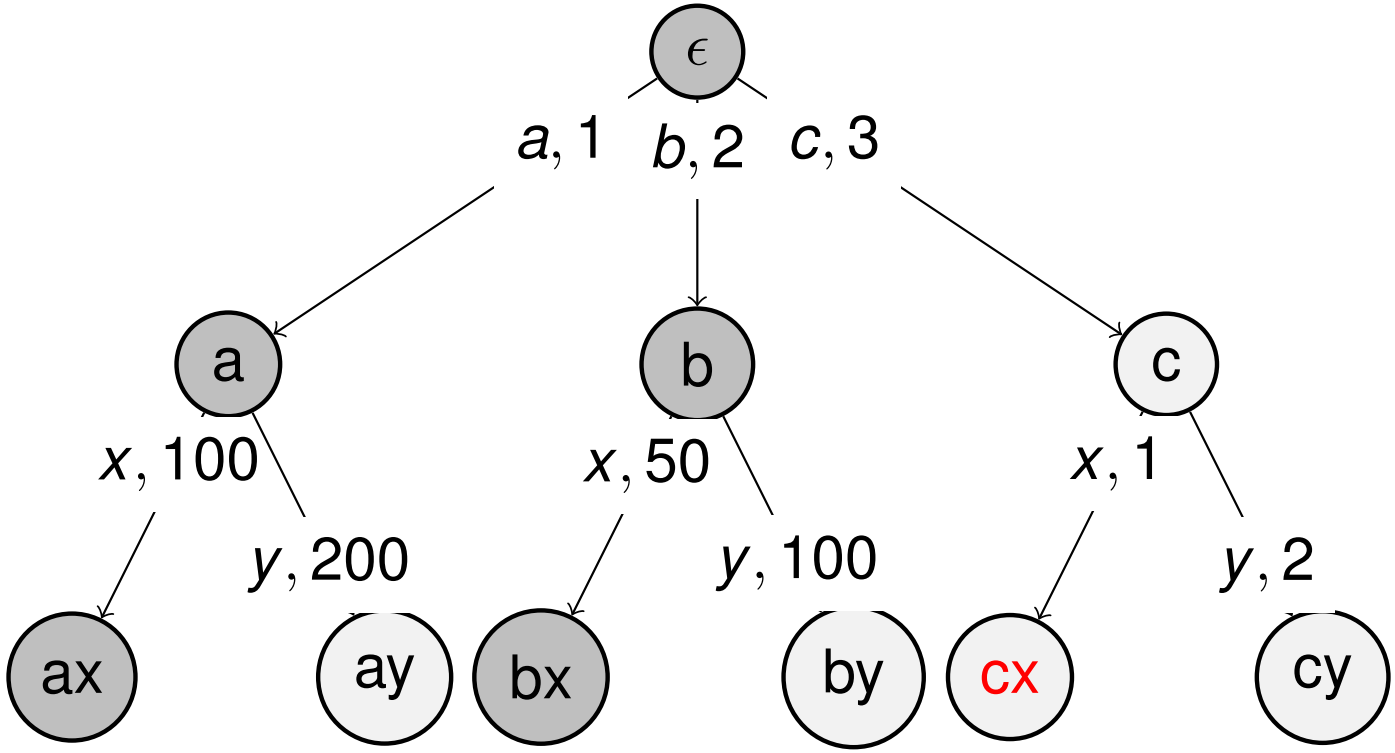


Figure : Beam search, beam size = 2. Traversed nodes are marked in grey.

# Beam search example

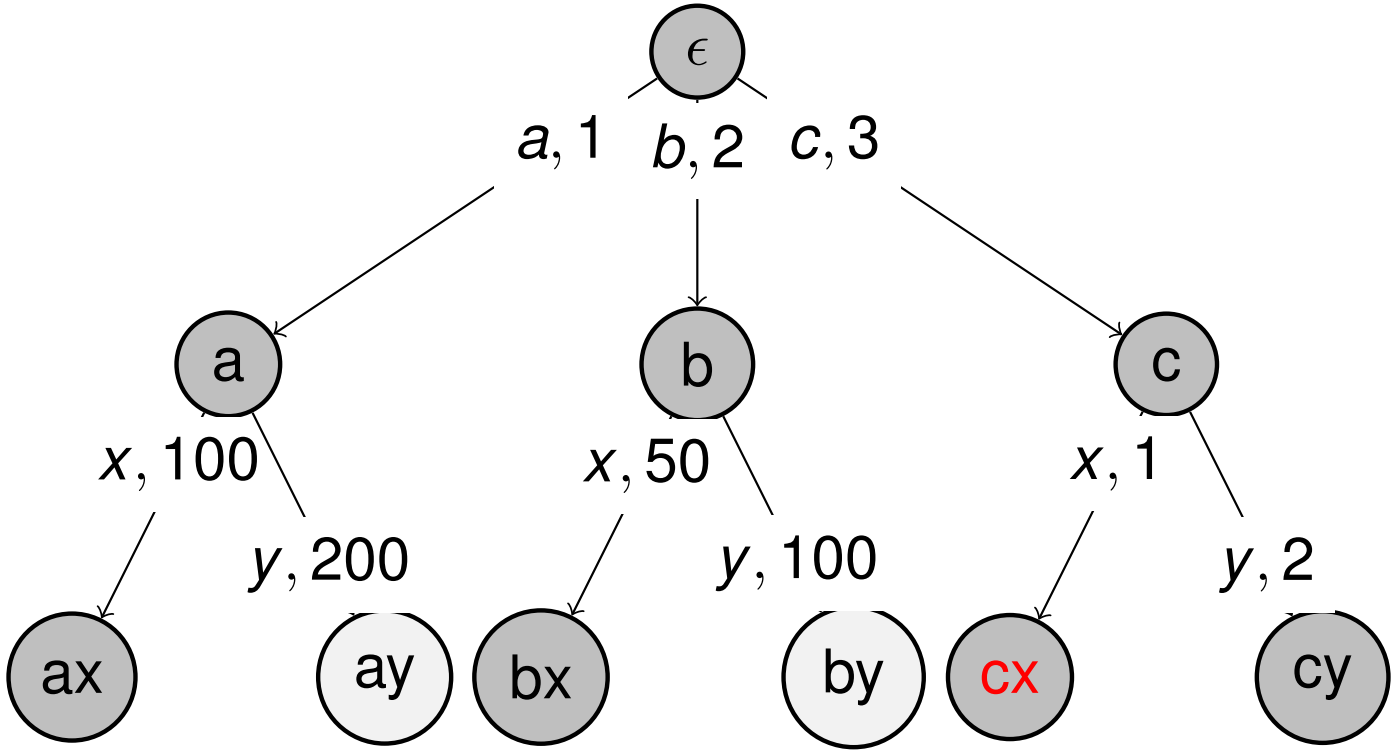


Figure : Beam search, beam size = 4. Traversed nodes are marked in grey.

# Outline

# Example

- at the gam es: sp inal i _njury, amputee blindness, andLes Autres.
- at the games: spinal i njury, amputee blin dness, and Les Autres.

$B =$‘ gam es: sp inal i’, $v =$‘ ’, $A =$‘ njury, amputee blin’, $s =$‘s’

| op | $X$ | $Y$ | $f(X, Y, 3)$ | $b(X, Y, 3)$ | $3^2 \cdot P_o(X, Y)$ | $\log 9P_o$ |
|---|---|---|---|---|---|---|
| DEL | $B$ | $A$ | 0.53 | 1.55 | 0.82 | **-0.19** |
| NOP | $Bv$ | $A$ | 0.06 | 0.0007 | $4 \cdot 10^{-5}$ | $-10.11$ |
| NOP | $B$ | $vA$ | 0.001 | 0.058 | $8 \cdot 10^{-5}$ | $-9.43$ |
| ADD | $Bs$ | $vA$ | 0.87 | 0.26 | 0.22 | $-1.49$ |
| ADD | $B$ | $svA$ | 0.63 | 0.60 | 0.38 | $-0.97$ |

# Outline

# Training Results

| Dir | Model | Loss | Acc | Top 5-Acc |
|---|---|---|---|---|
| back | BiLSTM128H20L1R | 1.503 | 0.565 | 0.838 |
| back | GRU128H20L2R | 1.469 | 0.576 | **0.844** |
| back | LSTM128H20L2N | 1.555 | 0.554 | 0.831 |
| back | LSTM128H20L2R | **1.454** | **0.577** | **0.844** |
| forw | BiLSTM128H20L1R | 1.504 | 0.566 | 0.840 |
| forw | GRU128H20L2R | 1.574 | 0.553 | 0.832 |
| forw | LSTM128H20L2N | **1.450** | **0.579** | **0.847** |
| forw | LSTM128H20L2R | 1.565 | 0.551 | 0.832 |

Table : Simple-Wikipedia test set loss and accuracies. The loss is categorical cross-entropy. The shown results are the 3-fold cross validated.

# Training Results

| Dir | Model | Loss | Acc | Top 5-Acc |
|---|---|---|---|---|
| back | BiLSTM64H20L1N | 1.670 | 0.519 | 0.822 |
| back | BiLSTM128H20L1N | 1.414 | 0.585 | 0.858 |
| back | BiLSTM128H40L1N | 1.429 | 0.579 | 0.853 |
| back | BiLSTM256H20L1N | **1.282** | **0.620** | **0.875** |
| back | BiLSTM256H40L1N | 1.368 | 0.602 | 0.862 |
| forw | BiLSTM64H20L1N | 1.649 | 0.534 | 0.828 |
| forw | BiLSTM128H20L1N | 1.419 | 0.594 | 0.857 |
| forw | BiLSTM128H40L1N | 1.427 | 0.588 | 0.855 |
| forw | BiLSTM256H20L1N | **1.282** | **0.629** | **0.874** |
| forw | BiLSTM256H40L1N | 1.355 | 0.613 | 0.864 |

Table : Reuters-21578 test set loss and accuracies. The loss is categorical cross-entropy. The shown results are the 3-fold cross validated.

# Outline

# Retokenization

Retokenization of a group tokens $T_1, \cdots, T_n$, which are joined into a big word $w$, is done by retokenizing any suffix of $w$ according to:

$$B_w[i] = \max_{i<j\leq|w|+1} \{B_w[j] \cdot \theta_{i,j} + (1 - \theta_{i,j}) \cdot Score(w_{i\rightarrow j})\} \tag{13}$$

Where $\theta_{i,j} = \frac{|w|-j+1}{|w|-i+1}$ is a normalizing term.

- $B_w[i]$ solves the suffix $w_{i\rightarrow}$
- Choose the first token ending at position $j$, using $Score(w_{i\rightarrow j})$
- Solve the remaining shorter suffix recursively, using $B_w[j]$

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| $j$ | - | - | - | - | - | - | 7 | 8 | 9 | 10 | 11 |
| $\theta_{i,j}$ | — | —- | — | — | — | - | 0.80 | 0.60 | 0.40 | 0.20 | 0.0 |
| $\theta_{i,j}B$ | — | —- | — | — | — | - | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $S(w_{i\to j})$ | — | —- | — | — | — | - | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 |
| $(1-\theta_{i,j})S(w_{i\to j})$ | — | —- | — | — | — | - | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 |
| $B$ | — | —- | — | — | — | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $nxt$ | - | - | - | - | - | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’.

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| $j$ | - | - | - | - | - | 6 | 7 | 8 | 9 | 10 | 11 |
| $\theta_{i,j}$ | — | —- | — | — | — | 0.83 | 0.67 | 0.50 | 0.33 | 0.17 | 0.0 |
| $\theta_{i,j}B$ | — | —- | — | — | — | 0.42 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $S(w_{i\to j})$ | — | —- | — | — | — | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $(1-\theta_{i,j})S(w_{i\to j})$ | — | —- | — | — | — | 0.08 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $B$ | — | —- | — | — | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $nxt$ | - | - | - | - | 6 | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’.

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| $j$ | - | - | - | - | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $\theta_{i,j}$ | — | —- | — | — | 0.86 | 0.71 | 0.57 | 0.43 | 0.29 | 0.14 | 0.0 |
| $\theta_{i,j}B$ | — | —- | — | — | 0.43 | 0.36 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $S(w_{i\to j})$ | — | —- | — | — | 0.0 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $(1-\theta_{i,j})S(w_{i\to j})$ | — | —- | — | — | 0.0 | 0.14 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $B$ | — | —- | — | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $nxt$ | - | - | - | 6 | 6 | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’.

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| $j$ | - | - | - | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $\theta_{i,j}$ | — | —- | — | 0.88 | 0.75 | 0.62 | 0.50 | 0.38 | 0.25 | 0.12 | 0.0 |
| $\theta_{i,j}B$ | — | —- | — | 0.44 | 0.38 | 0.31 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $S(w_{i\to j})$ | — | —- | — | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $(1-\theta_{i,j})S(w_{i\to j})$ | — | —- | — | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $B$ | — | —- | 0.44 | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $nxt$ | - | - | 4 | 6 | 6 | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’.

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| $j$ | - | - | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $\theta_{i,j}$ | —— | —- | 0.89 | 0.78 | 0.67 | 0.56 | 0.44 | 0.33 | 0.22 | 0.11 | 0.0 |
| $\theta_{i,j}B$ | —— | —- | 0.39 | 0.39 | 0.33 | 0.28 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $S(w_{i\to j})$ | —— | —- | 0.0 | 0.0 | 0.0 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 |
| $(1-\theta_{i,j})S(w_{i\to j})$ | —— | —- | 0.0 | 0.0 | 0.0 | 0.22 | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 |
| $B$ | —— | 0.5 | 0.44 | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $nxt$ | - | 6 | 4 | 6 | 6 | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’.

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| $j$ | - | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $\theta_{i,j}$ | —— | 0.90 | 0.80 | 0.70 | 0.60 | 0.50 | 0.40 | 0.30 | 0.20 | 0.10 | 0.0 |
| $\theta_{i,j}B$ | —— | 0.45 | 0.35 | 0.35 | 0.3 | 0.25 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $S(w_{i\to j})$ | —— | 0.0 | 0.0 | 0.0 | 0.5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $(1-\theta_{i,j})S(w_{i\to j})$ | —— | 0.0 | 0.0 | 0.0 | 0.2 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $B$ | 0.75 | 0.5 | 0.44 | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $nxt$ | 6 | 6 | 4 | 6 | 6 | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’.

# Retokenization example

| | H | e | l | l | o | w | a | r | l | d | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *i* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | - |
| *B* | 0.75 | 0.5 | 0.44 | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| *nxt* | 6 | 6 | 4 | 6 | 6 | 11 | 8 | 9 | 10 | 11 | - |

Table : Values of retokenization of ‘Hellowarld’ into ‘Hello world’. The grey cells mark the beginning of the new tokens according to *nxt*.

# Grouping

- Grouping $T_1, \cdots, T_n$ is done by grouping any suffix of tokens $T_i, \cdots, T_n$ according to:

$$F[i] := \max_{1 \leq d \leq \omega, n+1-i} \{F[i+d] + G(i,d)\} \tag{14}$$

Where $G$ retokenizes the group of chosen tokens $T_i, \cdots, T_{i+d-1}$:

$$G(i,d) = \max \begin{cases} \text{Score}(T_i \circ T_{i+1} \circ \cdots \circ T_{i+d-1}) & \text{if not 0} \\ \text{Retokenize}(T_i \circ T_{i+1} \circ \cdots \circ T_{i+d-1}) & \text{otherwise} \end{cases} \tag{15}$$

- The remaining suffix of tokens is solved recursively by $F[i+d]$.
- The group size is restricted up to $\omega \leq 8$ tokens.

# Grouping in action

| $i$ | $d$ | Tokens group | Retokenization | Joined | Score $G(i, d)$ |
|---|---|---|---|---|---|
| 1 | 1 | He | He | - | 0.0 |
| 1 | 2 | He, llowarl | Hello, w, a, r, l | - | 0.56 |
| **1** | **3** | He, llowarl, d | Hello, world | - | **0.75** |
| 2 | 1 | llowarl | l, to, w, a, r | - | 0.14 |
| **2** | **2** | llowarl, d | l, to, world | - | 0.44 |
| 2 | 3 | llowarl, d, td | l, to, world, to | - | **0.45** |
| 3 | 1 | d | d | - | 0.0 |
| 3 | 2 | d, td | d, to | - | 0.33 |
| **3** | **3** | d, td, ay | d, today | - | **0.4** |
| 4 | 1 | td | - | to | 0.5 |
| **4** | **2** | td, ay | - | today | **0.5** |
| **5** | **1** | ay | ay | - | **0.0** |

Table : $G(i, d)$ for the corrupt text: 'He$_1$ llowarl$_2$ d$_3$ td$_4$ ay$_5$'

# Grouping in action

| | | He | llowarl | d | td | ay |
|---|---|---|---|---|---|---|
| | $G$ | 1 | 2 | 3 | 4 | 5 |
| He | 1 | 0.0 | 0.56 | 0.75 | - | - |
| | 2 | - | 0.14 | 0.44 | 0.45 | - |
| | 3 | - | - | 0.0 | 0.33 | 0.4 |
| td | 4 | - | - | - | 0.5 | 0.5 |
| | 5 | - | - | - | - | 0.0 |

| | He | llowarl | d | td | ay |
|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 |
| $F$ | 1.25 | 0.94 | 0.5 | 0.5 | 0.0 |
| *nxts* | 4 | 4 | 4 | 6 | 6 |

Table : Functions $F[i]$ and $G(i, d)$ for the tokens. Grey cells mark the beginnings of the groups. The text is grouped into '(He$_1$ llowarl$_2$ d$_3$) (td$_4$ ay$_5$)'

# Outline

# Recurrent neural networks

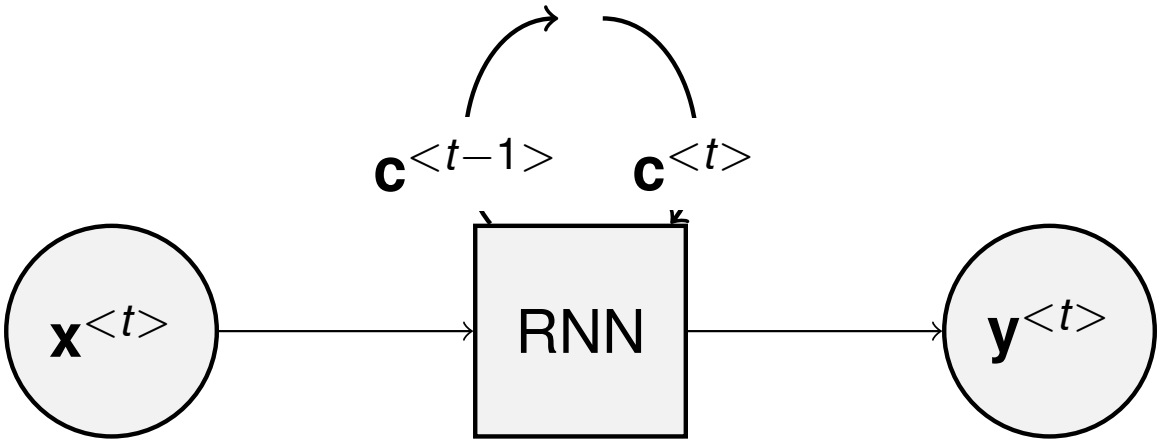


Figure : Recurrent neural network

- RNNs process sequences over time.
- LSTM and GRU are commonly used RNN architectures.
- LSTM and GRU ”remember” parts of the input.

# Outline

# Example

| Fixed text | precision | recall | $F_1$ |
|---|---|---|---|
| ‘Helloworld!’ | 0.75 | 0.75 | 0.75 |
| ‘Hello world!’ | 0.80 | 1.00 | 0.89 |
| ‘Helloworld’ | 1.00 | 0.75 | 0.86 |

Table : $G =$ ‘Hello world’, $C =$ ‘Halloword’