Adding new words into a language model using parameters of known words with similar behavior

Luiza Orosanu, Denis Jouvet

INRIA-Loria, Nancy, France
Multispeech Team
Summary

1. Context

2. Approach

3. Experiments

4. Conclusions and future work
Context

- language models in automatic speech recognition systems
- trained on large text data sets
- having closed vocabulary generating OOV problems

Our study

- add new words that are specific to a certain domain
- avoid recognition errors of words that are frequently pronounced (yet unknown by the system)
Our approach

- without retraining or adapting the model (which requires a lot of new data relative to the new words)

- approach based on the similarity with in-vocabulary words
  two words are similar if they appear in similar contexts
  \[\iff\text{they have similar neighbor distributions}\]
Summary

1 Context

2 Approach

3 Experiments

4 Conclusions and future work
Approach

- use a few examples of sentences for each new word
- find similar known words (similar neighbor distributions)
- transpose LM probabilities from similar words to new words
1. Acquire a few **examples of sentences** with the new word

\[ P_k(w|nW) \text{ for } k \in \{..., -2, -1, +1, +2, ...\} \]
1. Acquire a few examples of sentences with the new word
   → compute the neighbor distributions of the new word $nW$
   
   $P_k(w|nW)$ for $k \in \{\ldots, -2, -1, +1, +2, \ldots\}$

- example of new word: **soir**
- examples of sentences
  
  on ignorait encore lundi **soir** les conditions de sa survie
  devine qui vient dîner ce **soir**
  pas de consigne de vote au **soir** du premier tour

- preceding and following neighbors
  
<table>
<thead>
<tr>
<th>$k$</th>
<th>preceding and following neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>encore, dîner, vote, ...</td>
</tr>
<tr>
<td>-1</td>
<td>lundi, ce, au, ...</td>
</tr>
<tr>
<td>+1</td>
<td>les, du, ...</td>
</tr>
<tr>
<td>+2</td>
<td>conditions, premier, ...</td>
</tr>
</tbody>
</table>
2. Search for similar words in a reference corpus
   \[ P_k(w' | kW) \text{ for } k \in \{..., -2, -1, +1, +2, ...\} \]
2. Search for similar words in a reference corpus

→ compute the neighbor distributions of each known word \( kW \)

\[
P_k(w'|kW) \text{ for } k \in \{-2, -1, +1, +2, \ldots\}
\]

use directly the counts file of n-gram sequences

- \( 2g \Rightarrow \text{ maximum 2 neighbors } k \in \{-1, +1\} \)
- \( 3g \Rightarrow \text{ maximum 4 neighbors } k \in \{-2, -1, +1, +2\} \)

examples of 3-gram entries with the known word 'matin'

- "beau matin de 9" \( \rightarrow k=-1 \text{ neighbor } "beau"; k=+1 \text{ neighbor } "de" \)
- "matin a été 10" \( \rightarrow k=+1 \text{ neighbor } "a"; k=+2 \text{ neighbor } "été" \)
- "jusqu' au matin 28" \( \rightarrow k=-2 \text{ neighbor } "jusqu'"; k=-1 \text{ neighbor } "au" \)

preceeding and following neighbors

| \( k \) | 'jusqu', ...
| --- | ---
| -2 | beau, au, ...
| -1 | de, a, ...
| +1 | été, ...
| +2 |
3. Compute the **KL divergence** of neighbor distributions → between each known word ($kW$) and a new word ($nW$)

$$D_{KL}( P_k(\bullet|kW) \parallel P_k(\bullet|nW) ) = \sum_w P_k(w|kW) \log \left( \frac{P_k(w|kW)}{P_k(w|nW)} \right)$$

$$D(kW, nW) = \sum_k D_k(kW, nW)$$
3. Compute the **KL divergence** of neighbor distributions → between each known word ($kW$) and a new word ($nW$)

\[
D_{KL} \left( P_k(\cdot | kW) \ || \ P_k(\cdot | nW) \right) = \sum_w P_k(w | kW) \log \left( \frac{P_k(w | kW)}{P_k(w | nW)} \right)
\]

\[
D(kW, nW) = \sum_k D_k(kW, nW)
\]

4. Select the most similar words to a new word → those having minimal divergences
### Approach

- **Examples of similar words**

<table>
<thead>
<tr>
<th>Word</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>soir</td>
<td>matin, midi, dimanche, samedi, vendredi</td>
</tr>
<tr>
<td>soirs</td>
<td>temps, joueurs, matchs, pays, matches</td>
</tr>
<tr>
<td>année</td>
<td>époque, opération, expérience, épreuve, édition</td>
</tr>
<tr>
<td>années</td>
<td>décennies, saisons, épisodes, heures, opérations</td>
</tr>
<tr>
<td>gouvernement</td>
<td>parti, président, peuple, roi, mouvement</td>
</tr>
<tr>
<td>gouvernements</td>
<td>ministres, partis, syndicats, services, pays</td>
</tr>
<tr>
<td>guerre</td>
<td>campagne, crise, paix, position, ville</td>
</tr>
<tr>
<td>guerres</td>
<td>combats, opérations, missions, campagnes, séries</td>
</tr>
</tbody>
</table>
5. **Transpose the probabilities** of similar words onto new words

   → look for n-grams containing the similar words
   → replace the 'similar words' by the 'new word'
   → add the new n-grams into the new language model
5. **Transpose the probabilities** of similar words onto new words

→ look for n-grams containing the similar words
→ replace the 'similar words' by the 'new word'
→ add the new n-grams into the new language model

- new word "soir" similar to the known word "matin"

- **known n-grams** (in the language model)
  
  "-1.48214 possible ce matin"
  
  "-1.404164 matin ajoute que"

- **new n-grams** (to add in the new LM)
  
  "-1.48214 possible ce soir"
  
  "-1.404164 soir ajoute que"
Summary

1. Context
2. Approach
3. Experiments
   - Setup for experiments
   - Results
4. Conclusions and future work
1 Context

2 Approach

3 Experiments
   - Setup for experiments
   - Results

4 Conclusions and future work
Setup for experiments

- Select a list of new words to add to a language model
  ⇒ 44 new words

- Search for similar words
  - configuration
    - sentences based on "word|part-of-speech" units
    - 4 neighbor positions for each word: \( k \in \{-2, -1, +1, +2\} \)
    - choose the 10 most similar words for each new word

  - evaluate the impact of using \( \{5, 10, 20 \text{ or } 50\} \) examples of sentences for each new word

Sentences based on "word|part-of-speech" units

<table>
<thead>
<tr>
<th>qui</th>
<th>vient</th>
<th>dîner</th>
<th>ce</th>
<th>soir</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRO:REL</td>
<td>qui</td>
<td>VER:pres</td>
<td>vient</td>
<td>VER:infi</td>
</tr>
</tbody>
</table>
Setup for experiments

- The **BASELINE** language model
  * large vocabulary language model
  * trained by interpolation on various textual data
  * does not know the 44 new words

- The **ORACLE** language model
  * large vocabulary language model
  * trained by interpolation on various textual data
  * knows the 44 new words

<table>
<thead>
<tr>
<th></th>
<th>LM</th>
<th>1-grams</th>
<th>2-grams</th>
<th>3-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE</td>
<td>97 305</td>
<td>42.9M</td>
<td>79.2M</td>
<td></td>
</tr>
<tr>
<td>ORACLE</td>
<td>97 349</td>
<td>43.3M</td>
<td>80.1M</td>
<td></td>
</tr>
</tbody>
</table>
The **new** language models created

* by using \{5, 10, 20, 50\} examples of sentences for each new word
* by adding 1-grams or 1-,2-,3-grams of new words into the BASELINE LM

<table>
<thead>
<tr>
<th></th>
<th>baseline</th>
<th>new language models</th>
<th>ORACLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5ex</td>
<td>10ex</td>
</tr>
<tr>
<td># 2-grams</td>
<td>42.9</td>
<td>44.7</td>
<td>44.6</td>
</tr>
<tr>
<td># 3-grams</td>
<td>79.2</td>
<td>89.8</td>
<td>89.3</td>
</tr>
</tbody>
</table>

**Table**: Number [in millions] of 2-grams and 3-grams in the new 'baseline+1-,2-,3-grams' LMs

⇒ The new 'baseline+1-,2-,3-grams' adds:

* between 1.7M and 1.9M new 2-grams
* between 10.6M and 11.6M new 3-grams
Setup for experiments

- Setup for evaluations
  - the LMs are evaluated over the ESTER2 development set
  - the 44 new words have an occurrence frequency of 1.33%

- Compare the performance of new LMs with baseline LM
  - regarding the WER
  - regarding the percentage of new words correctly recognized
Summary

1. Context

2. Approach

3. Experiments
   - Setup for experiments
   - Results

4. Conclusions and future work
Results: the WER performances

<table>
<thead>
<tr>
<th></th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE</td>
<td>26.79%</td>
</tr>
<tr>
<td>ORACLE</td>
<td>24.79%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># examples of sentences</th>
<th>5ex</th>
<th>10ex</th>
<th>20ex</th>
<th>50ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline+1-grams</td>
<td>26.45</td>
<td>26.44</td>
<td>26.40</td>
<td>26.42</td>
</tr>
<tr>
<td>baseline+1-,2-,3-grams</td>
<td>25.68</td>
<td><strong>25.51</strong></td>
<td><strong>25.51</strong></td>
<td>25.57</td>
</tr>
</tbody>
</table>

**Table:** WER of new 'baseline+1-grams’ and 'baseline+1-,2-,3-grams’ LMs
Results: the WER performances

<table>
<thead>
<tr>
<th># examples of sentences</th>
<th>5ex</th>
<th>10ex</th>
<th>20ex</th>
<th>50ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline+1-grams</td>
<td>26.45</td>
<td>26.44</td>
<td>26.40</td>
<td>26.42</td>
</tr>
<tr>
<td>baseline+1-,2-,3-grams</td>
<td>25.68</td>
<td><strong>25.51</strong></td>
<td><strong>25.51</strong></td>
<td>25.57</td>
</tr>
</tbody>
</table>

Table: WER of new 'baseline+1-grams' and 'baseline+1-,2-,3-grams' LMs

⇒ adding only 1-grams for new words hardly improves the performance
⇒ adding 1-,2-,3-grams for new words provides results closer to the ORACLE’s performance
⇒ between 1.1% and 1.3% WER absolute reduction (compared to the baseline LM)
⇒ 0.7% WER difference with the ORACLE model
Results: percentage of new words correctly recognized

BASELINE 0.00%
ORACLE 85.45%

<table>
<thead>
<tr>
<th># examples of sentences</th>
<th>5ex</th>
<th>10ex</th>
<th>20ex</th>
<th>50ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline+1-grams</td>
<td>29.81</td>
<td>20.00</td>
<td>22.18</td>
<td>20.36</td>
</tr>
<tr>
<td>baseline+1-,2-,3-grams</td>
<td>60.54</td>
<td>61.81</td>
<td>64.90</td>
<td>62.76</td>
</tr>
</tbody>
</table>

Table: Correct recognition of new words of new 'baseline+1-grams' and 'baseline+1-,2-,3-grams' LMs
Results: percentage of new words correctly recognized

<table>
<thead>
<tr>
<th></th>
<th>0.00%</th>
<th>85.45%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORACLE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># examples of sentences</th>
<th>5ex</th>
<th>10ex</th>
<th>20ex</th>
<th>50ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline+1-grams</td>
<td>29.81</td>
<td>20.00</td>
<td>22.18</td>
<td>20.36</td>
</tr>
<tr>
<td>baseline+1-,2-,3-grams</td>
<td>60.54</td>
<td>61.81</td>
<td>64.90</td>
<td>62.76</td>
</tr>
</tbody>
</table>

Table: Correct recognition of new words of new 'baseline+1-grams' and 'baseline+1-,2-,3-grams' LMs

⇒ up to 65% of the new words correctly recognized
Summary

1. Context
2. Approach
3. Experiments
4. Conclusions and future work
Conclusions and future work

Conclusions

* adding only 1-grams for new words hardly improves the performance
* adding 1-,2-,3-grams for new words provides results closer to the ORACLE’s performance
* the similarity approach and the proposed method to add new n-grams into a language model are efficient

Investigate further

* the setups for finding similar words
* filter the n-grams of new words (diminish the size of new LMs)
* consider a different number of similar words for each new word
Thank you for your attention!
1. acquire a few **examples of sentences** with the new word

→ compute the neighbor distributions of the new word \( nW \)

\[
P_k(w|nW) \quad \text{for} \quad k = \{...,-3,-2,-1,+1,+2,+3,...\}
\]

- example of new word: **tournevis**
- examples of sentences
  - le **tournevis** motorisé s’ appelle une visseuse
  - un **tournevis** suffit pour le démontage
  - l’ embout du **tournevis** peut vriller si on serre trop fort
  - la tête du **tournevis** peut être plate cruciforme ou autre
  - ...

- preceeding and following neighbors

  | p-2 [#13] | (tête #1) (embout #1) (a #1) (manche #1) (poignées #1) ...
  | p-1 [#4]  | (le #6) (un #6) (du #3) (de #3)
  | p+1 [#13] | (motorisé #1) (suffit #1) (peut #2) (et #2) (standard #1) (cruciforme #1)(plat #1) ...
  | p+2 [#13] | (s’ #1) (pour #1) (peut #1) (être #1) (aux #1) (est #1) (exercer #1) (un #2) (vriller #1) ...

- the [p-1] neighbor distribution of new word ”**tournevis**”

<table>
<thead>
<tr>
<th>nW=tournevis</th>
<th>[p-1]</th>
<th>le</th>
<th>un</th>
<th>du</th>
<th>de</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#18</td>
<td>0.333</td>
<td>0.333</td>
<td>0.167</td>
<td>0.167</td>
</tr>
</tbody>
</table>
2. use **reference corpus** to search for similar words
   → compute the neighbor distributions of each known word $kW$
   \[
P_k(w' | kW) \text{ for } k = \{-3, -2, -1, +1, +2, +3, \ldots\}\]

- use directly the counts file of n-gram sequences
  * $2g \Rightarrow$ maximum 2 neighbors [p-1], [p+1]
  * $3g \Rightarrow$ maximum 4 neighbors [p-2], [p-1], [p+1], [p+2]

- an exemple of a 3-gram entry: "du monde numérique 3"
  * the known word "monde"
    → previous neighbor [p-1] : du
    → following neighbor [p+1] : numérique

- preceeding and following neighbors of word "monde"
  \[
  \begin{array}{c|c}
  p-2 [#685] & (page #1) \text{ (ordre #1) (mettre 1) (mais #2) (souvent 1) (exemple 1) (a 3) (tête 1) \ldots} \\
  p-1 [#40] & (du #601) \text{ (quart 11) (autre #12) (un #108) (le #531) (de #54) \ldots} \\
  p+1 [#531] & (numérique #4) \text{ (cherchent #3) (virtuel #8) (est #63) (va #11) (et #73) \ldots} \\
  p+2 [#860] & (chrétien #1) \text{ (doit #4) (surtout #1) (pour #8) (un #15) (être #9) (une #12) \ldots} \\
  \end{array}
  \]

- the [p-1] neighbor distribution of known word "monde"
  \[
  \begin{array}{c|c|c|c|c}
  kW=monde & [p-1] & le & un & du & de \\
  \hline
  & #1724 & 0.308 & 0.063 & 0.349 & 0.031 \\
  \end{array}
  \]
3. compute the **KL divergence** between the neighbor distributions of all known word ($kW$) and a new word ($nW$)

$$D_{KL} \left( P_k(\bullet|kW) \parallel P_k(\bullet|nW) \right) = \sum_w P_k(w|kW) \log \left( \frac{P_k(w|kW)}{P_k(w|nW)} \right)$$

compute the divergence between the [p-1] neighbor distributions

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>le</th>
<th>un</th>
<th>du</th>
<th>de</th>
</tr>
</thead>
<tbody>
<tr>
<td>$nW=\text{tournevis}$</td>
<td>0.333</td>
<td>0.333</td>
<td>0.167</td>
<td>0.167</td>
</tr>
<tr>
<td>$kW=\text{monde}$</td>
<td>0.308</td>
<td>0.063</td>
<td>0.349</td>
<td>0.031</td>
</tr>
</tbody>
</table>
3. compute the KL divergence between the neighbor distributions of all known word \((kW)\) and a new word \((nW)\)

\[
D_{KL} ( P_k(\cdot|kW) \mid\mid P_k(\cdot|nW) ) = \sum_w P_k(w|kW) \log \left( \frac{P_k(w|kW)}{P_k(w|nW)} \right)
\]

- compute the divergence between the \([p-1]\) neighbor distributions

<table>
<thead>
<tr>
<th>([p-1])</th>
<th>le</th>
<th>un</th>
<th>du</th>
<th>de</th>
</tr>
</thead>
<tbody>
<tr>
<td>kW=monde</td>
<td>0.308</td>
<td>0.063</td>
<td>0.349</td>
<td>0.031</td>
</tr>
<tr>
<td>nW=tournevis</td>
<td>0.333</td>
<td>0.333</td>
<td>0.167</td>
<td>0.167</td>
</tr>
</tbody>
</table>

\[
D(w) = P(w|kW) \log_2 \left( \frac{P(w|kW)}{P(w|nW)} \right)
\]

\[
D(w) = 0.308 \log_2 \left( \frac{0.308}{0.333} \right)
\]

\[
D(w) = -0.035
\]
3. compute the KL divergence between the neighbor distributions of all known word (kW) and a new word (nW)

\[ D_{KL}(P_k(\bullet|kW) \parallel P_k(\bullet|nW)) = \sum_w P_k(w|kW) \log \left( \frac{P_k(w|kW)}{P_k(w|nW)} \right) \]

compute the divergence between the [p-1] neighbor distributions

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>le</th>
<th>un</th>
<th>du</th>
<th>de</th>
</tr>
</thead>
<tbody>
<tr>
<td>nW=tournevis</td>
<td>0.333</td>
<td>0.333</td>
<td>0.167</td>
<td>0.167</td>
</tr>
<tr>
<td>kW=monde</td>
<td>0.308</td>
<td>0.063</td>
<td>0.349</td>
<td>0.031</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>(D(le))</th>
<th>(D(un))</th>
</tr>
</thead>
<tbody>
<tr>
<td>kW=monde,nW=tournevis</td>
<td>-0.035</td>
<td>-0.151</td>
</tr>
</tbody>
</table>

\[ w = un \]

\[ D(w) = P(w|kW) \log_2 \left( \frac{P(w|kW)}{P(w|nW)} \right) \]

\[ D(w) = 0.063 \log_2 \left( \frac{0.063}{0.333} \right) \]

\[ D(w) = -0.151 \]
3. compute the **KL divergence** between the neighbor distributions of all known word \((kW)\) and a new word \((nW)\)

\[
D_{KL}( P_k(\bullet|kW) \parallel P_k(\bullet|nW) ) = \sum_w P_k(w|kW) \log \left( \frac{P_k(w|kW)}{P_k(w|nW)} \right)
\]

compute the divergence between the \([p-1]\) neighbor distributions

\[
\begin{array}{|c|c|c|c|c|}
\hline
[p-1] & le & un & du & de \\
\hline
nW=tournevis & 0.333 & 0.333 & 0.167 & 0.167 \\
kW=monde & 0.308 & 0.063 & 0.349 & 0.031 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
[p-1] & D(le) & D(un) & D(du) \\
\hline
kW=monde,nW=tournevis & -0.035 & -0.151 & 0.371 \\
\hline
\end{array}
\]

\[
w = du
\]

\[
D(w) = P(w|kW) \log_2 \left( \frac{P(w|kW)}{P(w|nW)} \right)
\]

\[
D(w) = 0.349 \log_2 \left( \frac{0.349}{0.167} \right)
\]

\[
D(w) = 0.371
\]
3. compute the **KL divergence** between the neighbor distributions of all known word ($kW$) and a new word ($nW$)

\[
D_{KL}( P_k(\bullet|kW) \parallel P_k(\bullet|nW) ) = \sum_{w} P_k(w|kW) \log \left( \frac{P_k(w|kW)}{P_k(w|nW)} \right)
\]

- compute the divergence between the [p-1] neighbor distributions

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>le</th>
<th>un</th>
<th>du</th>
<th>de</th>
</tr>
</thead>
<tbody>
<tr>
<td>nW=tournevis</td>
<td>0.333</td>
<td>0.333</td>
<td>0.167</td>
<td>0.167</td>
</tr>
<tr>
<td>kW=monde</td>
<td>0.308</td>
<td>0.063</td>
<td>0.349</td>
<td>0.031</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>$D(le)$</th>
<th>$D(un)$</th>
<th>$D(du)$</th>
<th>$D(de)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>kW=monde,nW=tournevis</td>
<td>-0.035</td>
<td>-0.151</td>
<td>0.371</td>
<td><strong>-0.075</strong></td>
</tr>
</tbody>
</table>

**w = de**

\[
D(w) = P(w|kW) \log_2 \left( \frac{P(w|kW)}{P(w|nW)} \right)
\]

\[
D(w) = 0.031 \log_2 \left( \frac{0.031}{0.167} \right)
D(w) = -0.075
\]
Annexe

3. compute the KL divergence between the neighbor distributions of all known word ($kW$) and a new word ($nW$)

$$D_{KL}(P_k(\bullet|kW) \parallel P_k(\bullet|nW)) = \sum_w P_k(w|kW) \log \left(\frac{P_k(w|kW)}{P_k(w|nW)}\right)$$

- compute the divergence between the [p-1] neighbor distributions

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>le</th>
<th>un</th>
<th>du</th>
<th>de</th>
</tr>
</thead>
<tbody>
<tr>
<td>$nW=\text{tournevis}$</td>
<td>0.333</td>
<td>0.333</td>
<td>0.167</td>
<td>0.167</td>
</tr>
<tr>
<td>$kW=\text{monde}$</td>
<td>0.308</td>
<td>0.063</td>
<td>0.349</td>
<td>0.031</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[p-1]</th>
<th>$D(le)$</th>
<th>$D(un)$</th>
<th>$D(du)$</th>
<th>$D(de)$</th>
<th>$D_{KL}(p\parallel q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kW=\text{monde}, nW=\text{tournevis}$</td>
<td>-0.035</td>
<td>-0.151</td>
<td>0.371</td>
<td>-0.075</td>
<td>0.110</td>
</tr>
</tbody>
</table>
4. select the most similar words to a new word with respect to the KL divergences

\[ D(kW, nW) = \sum_k D_k(kW, nW) \]

Example: Wikipedia corpus, 3-grams

<table>
<thead>
<tr>
<th>new word - known word</th>
<th>Total</th>
<th>[p-2]</th>
<th>[p-1]</th>
<th>[p+1]</th>
<th>[p+2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>tournevis-système</td>
<td>26.8459</td>
<td>10.6792</td>
<td>0.7044</td>
<td>7.9961</td>
<td>7.4662</td>
</tr>
<tr>
<td>tournevis-jeu</td>
<td>27.6926</td>
<td>10.6276</td>
<td>0.4276</td>
<td>9.0107</td>
<td>7.6267</td>
</tr>
<tr>
<td>tournevis-modèle</td>
<td>28.3001</td>
<td>11.5795</td>
<td>0.7768</td>
<td>8.0591</td>
<td>7.8847</td>
</tr>
<tr>
<td>tournevis-véhicule</td>
<td>28.482</td>
<td>12.411</td>
<td>0.706</td>
<td>8.2637</td>
<td>7.1013</td>
</tr>
<tr>
<td>tournevis-traitement</td>
<td>29.0743</td>
<td>11.7698</td>
<td>0.5091</td>
<td>8.8681</td>
<td>7.9273</td>
</tr>
<tr>
<td>tournevis-courant</td>
<td>29.3598</td>
<td>10.9703</td>
<td>1.2209</td>
<td>9.1505</td>
<td>8.0181</td>
</tr>
<tr>
<td>tournevis-type</td>
<td>29.499</td>
<td>11.1394</td>
<td>1.445</td>
<td>8.6027</td>
<td>8.3119</td>
</tr>
<tr>
<td>tournevis-poisson</td>
<td>29.5627</td>
<td>11.9312</td>
<td>0.5941</td>
<td>9.6137</td>
<td>7.4237</td>
</tr>
<tr>
<td>tournevis-style</td>
<td>29.6316</td>
<td>11.3593</td>
<td>0.6154</td>
<td>9.5178</td>
<td>8.1391</td>
</tr>
<tr>
<td>tournevis-dispositif</td>
<td>29.6418</td>
<td>12.0949</td>
<td>0.8052</td>
<td>9.2124</td>
<td>7.5293</td>
</tr>
</tbody>
</table>
Annexe: Add a new word nW into a language model

1: newLM ← LM
2: newNgrams ← ∅
3: # process the reference ngrams
4: for each ngram ∈ LM do
5:   for each kW ∈ similarWords(nW) do
6:     if contains(ngram, kW) then
7:        ngram' ← replace(ngram, kW, nW)
8:        push(newNgrams, ngram')
9:     end if
10: end for
11: end for
12: # choose the new ngrams to add to the newLM
13: S ← getUniqueSequences(newNgrams)
14: for each seq ∈ S do
15:   if frequency(seq) = 1 then
16:     prob ← getProbability(seq)
17:   else
18:     P ← getProbabilities(seq)
19:     prob ← medianProbability(P)
20:   end if
21: push(newLM, “prob seq”)
22: end for