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

Context

- 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
    $\iff$ they have similar neighbor distributions

Summary

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

Approach

1. Acquire a few **examples of sentences** with the new word

   $\rightarrow$ compute the neighbor distributions of the new word nW

$$P_k(w|\mathbf{nW}) \text{ for } k \in \{..., -2, -1, +1, +2, ...\}$$

1. Acquire a few **examples of sentences** with the new word

$\rightarrow$ compute the neighbor distributions of the new word nW

$$P_k(w|\mathbf{nW}) \text{ for } k \in \{..., -2, -1, +1, +2, ...\}$$

- 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

- preceeding and following neighbors

k= -2	encore, dîner, vote, ...
k= -1	lundi, ce, au, ...
k= +1	les, du, ...
k= +2	conditions, premier, ...

Approach

2. Search for similar words in a **reference corpus**

$\rightarrow$ compute the neighbor distributions of each known word kW

$$P_k(w'|\mathbf{kW}) \text{ for } k \in \{..., -2, -1, +1, +2, ...\}$$

Approach

2. Search for similar words in a **reference corpus**

$\rightarrow$ compute the neighbor distributions of each known word kW

$$P_k(w'|\mathbf{kW}) \text{ for } k \in \{..., -2, -1, +1, +2, ...\}$$

- use directly the counts file of n-gram sequences
  - * 2g $\Rightarrow$ maximum 2 neighbors $k \in \{-1, +1\}$
  - * 3g $\Rightarrow$ 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 neighbor "beau"; k=+1 neighbor "de"
  - * `"`**`matin`** `a été 10"` $\rightarrow$ k=+1 neighbor "a"; k=+2 neighbor "été"
  - * `"jusqu' au` **`matin`** `28"` $\rightarrow$ k=-2 neighbor "jusqu'"; k=-1 neighbor "au"
- preceeding and following neighbors

k= -2	jusqu', ...
k= -1	beau, au, ...
k= +1	de, a, ...
k= +2	été, ...

Approach

3. Compute the **KL divergence** of neighbor distributions
$\rightarrow$ between each known word (kW) and a new word (nW)

$$D_{KL}\left(\; P_k(\bullet|\mathbf{kW}) \;||\; P_k(\bullet|\mathbf{nW}) \;\right) = \sum_{w} \; P_k(w|\mathbf{kW}) \; log\left(\frac{P_k(w|\mathbf{kW})}{P_k(w|\mathbf{nW})}\right)$$

$$D(\mathbf{kW}, \mathbf{nW}) = \sum_{k} D_k(\mathbf{kW}, \mathbf{nW})$$

Approach

3. Compute the **KL divergence** of neighbor distributions
$\rightarrow$ between each known word (kW) and a new word (nW)

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

$$D(\mathbf{kW}, \mathbf{nW}) = \sum_k D_k(\mathbf{kW}, \mathbf{nW})$$

4. Select **the most similar words** to a new word
$\rightarrow$ those having minimal divergences

Approach

$\Rightarrow$ examples of similar words

soir	$\rightarrow$ matin, midi, dimanche, samedi, vendredi
soirs	$\rightarrow$ temps, joueurs, matchs, pays, matches
année	$\rightarrow$ époque, opération, expérience, épreuve, édition
années	$\rightarrow$ décennies, saisons, épisodes, heures, opérations
gouvernement	$\rightarrow$ parti, président, peuple, roi, mouvement
gouvernements	$\rightarrow$ ministres, partis, syndicats, services, pays
guerre	$\rightarrow$ campagne, crise, paix, position, ville
guerres	$\rightarrow$ combats, opérations, missions, campagnes, séries

Approach

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

Approach

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

Summary

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
  $\Rightarrow$ **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 or 50} examples of sentences for each new word

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

qui	vient	dîner	ce	soir
PRO:REL|qui	VER:pres|vient	VER:infi|dîner	PRO:DEM|ce	NOM|soir

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

LM	1-grams	2-grams	3-grams
BASELINE	97 305	42.9M	79.2M
ORACLE	97 349	43.3M	80.1M

Setup for experiments

- 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

	baseline	**new language models**				**ORACLE**
		5ex	10ex	20ex	50ex	
# 2-grams	42.9	44.7	44.6	44.8	44.8	43.3
# 3-grams	79.2	89.8	89.3	90.5	90.8	80.1

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

$\Rightarrow$ 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

Results: the **WER** performances

BASELINE **26.79%**
ORACLE **24.79%**

	# examples of sentences			
	5ex	10ex	20ex	50ex
baseline+1-grams	26.45	26.44	26.40	26.42
baseline+1-,2-,3-grams	25.68	25.51	25.51	25.57

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

Results: the **WER** performances

BASELINE **26.79%**
ORACLE **24.79%**

	# examples of sentences			
	5ex	10ex	20ex	50ex
baseline+1-grams	26.45	26.44	26.40	26.42
baseline+1-,2-,3-grams	25.68	25.51	25.51	25.57

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%**

	# examples of sentences			
	5ex	10ex	20ex	50ex
baseline+1-grams	29.81	20.00	22.18	20.36
baseline+1-,2-,3-grams	60.54	61.81	64.90	62.76

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

BASELINE **0.00%**
ORACLE **85.45%**

	# examples of sentences			
	5ex	10ex	20ex	50ex
baseline+1-grams	29.81	20.00	22.18	20.36
baseline+1-,2-,3-grams	60.54	61.81	64.90	62.76

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

$\Rightarrow$ up to 65% of the new words correctly recognized

Summary

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 !

Annexe

1. aquire a few **examples of sentences** with the new word

   $\rightarrow$ compute the neighbor distributions of the new word nW

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

- 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"

	[p-1]	le	un	du	de
nW=tournevis	#18	0.333	0.333	0.167	0.167

Annexe

2. use **reference corpus** to search for similar words

   $\rightarrow$ compute the neighbor distributions of each known word kW

$$P_k(w'|\mathbf{kW}) \text{ for } k = \{..., -3, -2, -1, +1, +2, +3, ...\}$$

- 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"

    $\longrightarrow$ previous neighbor [p-1] : du

    $\longrightarrow$ following neighbor [p+1] : numérique
- preceeding and following neighbors of word "monde"

p-2 [#685]	(page #1) (ordre #1) (mettre 1) (mais #2) (souvent 1) (exemple 1) (a 3) (tête 1) ...
p-1 [#40]	(du #601) (quart 11) (autre #12) (un #108) (le #531) (de #54) ...
p+1 [#531]	(numérique #4) (cherchent #3) (virtuel #8) (est #63) (va #11) (et #73) ...
p+2 [#860]	(chrétien #1) (doit #4) (surtout #1) (pour #8) (un #15) (être #9) (une #12) ...

- the [p-1] neighbor distribution of known word "monde"

	[p-1]	le	un	du	de
kW=monde	#1724	0.308	0.063	0.349	0.031

Annexe

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

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

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

[p-1]	le	un	du	de
nW=tournevis	0.333	0.333	0.167	0.167
kW=monde	0.308	0.063	0.349	0.031

Annexe

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

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

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

[p-1]	le	un	du	de
nW=tournevis	0.333	0.333	0.167	0.167
kW=monde	0.308	0.063	0.349	0.031

[p-1]	$D(le)$
kW=monde,nW=tournevis	**-0.035**

w = le

$$D(w) = P\left(w|kW\right)log_2\left(\frac{P(w|kW)}{P(w|nW)}\right)$$
$$D(w) = 0.308 log_2\left(\frac{0.308}{0.333}\right)$$
$$D(w) = -0.035$$

Annexe

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

$$D_{KL}\left(\, P_k(\bullet|\mathbf{kW}) \;||\; P_k(\bullet|\mathbf{nW})\,\right) = \sum_{w} P_k(w|\mathbf{kW})\; log\left(\frac{P_k(w|\mathbf{kW})}{P_k(w|\mathbf{nW})}\right)$$

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

[p-1]	le	un	du	de
nW=tournevis	0.333	0.333	0.167	0.167
kW=monde	0.308	0.063	0.349	0.031

[p-1]	$D(le)$	$D(un)$
kW=monde,nW=tournevis	-0.035	**-0.151**

w = un

$$D(w) = P\left(w|kW\right) log_2\left(\frac{P(w|kW)}{P(w|nW)}\right)$$
$$D(w) = 0.063 log_2\left(\frac{0.063}{0.333}\right)$$
$$D(w) = -0.151$$

Annexe

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

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

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

[p-1]	le	un	du	de
nW=tournevis	0.333	0.333	0.167	0.167
kW=monde	0.308	0.063	0.349	0.031

[p-1]	$D(le)$	$D(un)$	$D(du)$
kW=monde,nW=tournevis	-0.035	-0.151	**0.371**

w = du

$$D(w) = P\,(w|kW)log_2\left(\frac{P(w|kW)}{P(w|nW)}\right)$$
$$D(w) = 0.349 log_2\left(\frac{0.349}{0.167}\right)$$
$$D(w) = 0.371$$

Annexe

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

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

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

[p-1]	le	un	du	de
nW=tournevis	0.333	0.333	0.167	0.167
kW=monde	0.308	0.063	0.349	0.031

[p-1]	$D(le)$	$D(un)$	$D(du)$	$D(de)$
kW=monde,nW=tournevis	-0.035	-0.151	0.371	**-0.075**

w = de

$$D(w) = P\left(w|kW\right) log_2\left(\frac{P(w|kW)}{P(w|nW)}\right)$$
$$D(w) = 0.031 log_2\left(\frac{0.031}{0.167}\right)$$
$$D(w) = -0.075$$

Annexe

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

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

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

[p-1]	le	un	du	de
nW=tournevis	0.333	0.333	0.167	0.167
kW=monde	0.308	0.063	0.349	0.031

[p-1]	$D(le)$	$D(un)$	$D(du)$	$D(de)$	$D_{KL}(p\|\|q)$
kW=monde,nW=tournevis	-0.035	-0.151	0.371	-0.075	**0.110**

Annexe

4. select **the most similar words** to a new word with respect to the KL divergences

$$D(\mathbf{kW}, \mathbf{nW}) = \sum_k D_k(\mathbf{kW}, \mathbf{nW})$$

Example : Wikipedia corpus, 3-grams

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

Annexe: Add a new word nW into a language model

```
newLM ← LM
newNgrams ← ∅
# process the reference ngrams
for each ngram ∈ LM do
    for each kW ∈ similarWords(nW) do
        if contains(ngram, kW) then
            ngram' ← replace(ngram, kW, nW)
            push(newNgrams, ngram')
        end if
    end for
end for
# choose the new ngrams to add to the newLM
S ← getUniqueSequences(newNgrams)
for each seq ∈ S do
    if frequency(seq) = 1 then
        prob ← getProbability(seq)
    else
        P ← getProbabilities(seq)
        prob ← medianProbability(P)
    end if
    push(newLM, "prob seq")
end for
```