Perplexity of n-gram and dependency language models

Martin Popel, David Mareček
ÚFAL, Charles University in Prague

TSD, 13th International Conference on Text, Speech and Dialogue
September 8, 2010, Brno

Outline

- Language Models (LM)
  - basics
  - design decisions
- Post-ngram LM
- Dependency LM
- Evaluation
- Conclusion & future plans

Language Models – basics

P(s) = ?

P(The dog barked again)

Language Models – basics

P(s) = ?

P(The dog barked again) > P(The dock barked again)

Language Models – basics

$P(s) = P(w_1, w_2, \ldots w_m)$

P(The dog barked again) =

$P(w_1=$ The $, w_2=$ dog $, w_3=$ barked $, w_4=$ again $)$

Language Models – basics

$$P(s) = P(w_1, w_2, \ldots w_m) = P(w_1)\, P(w_2|w_1) \ldots P(w_m|w_1,\ldots,w_{m-1})$$

Chain rule

P(The dog barked again) =

$P(w_1=$ The $)\cdot$

$P(w_2=$ dog $| w_1=$ The $)\cdot$

$P(w_3=$ barked $| w_1=$ The , $w_2=$ dog $)\cdot$

$P(w_4=$ again $| w_1=$ The , $w_2=$ dog , $w_3=$ barked $)$

Language Models – basics

$P(s) = P(w_1, w_2, \ldots w_m) = P(w_1)\, P(w_2|w_1) \ldots P(w_m|w_1,\ldots,w_{m-1})$

Changed notation

P(The dog barked again) =

$P(w_i =$ The $| i=1) \cdot$

$P(w_i =$ dog $| i=2, w_{i-1} =$ The $) \cdot$

$P(w_i =$ barked $| i=3, w_{i-2} =$ The $, w_{i-1} =$ dog $) \cdot$

$P(w_i =$ again $| i=4, w_{i-3} =$ The $, w_{i-2} =$ dog $, w_{i-1} =$ barked $)$

Language Models – basics

$P(s) = P(w_1, w_2, \dots w_m) = P(w_1)\, P(w_2|w_1) \dots P(w_m|w_1,\dots,w_{m-1})$

Artificial start-of-sentence token

$P(\boxed{\text{The dog barked again}}) =$

$P(w_i = \boxed{\text{The}} \mid i=1, w_{i-1} = \boxed{\text{NONE}}) \cdot$

$P(w_i = \boxed{\text{dog}} \mid i=2, w_{i-2} = \boxed{\text{NONE}}, w_{i-1} = \boxed{\text{The}}) \cdot$

$P(w_i = \boxed{\text{barked}} \mid i=3, w_{i-3} = \boxed{\text{NONE}}, w_{i-2} = \boxed{\text{The}}, w_{i-1} = \boxed{\text{dog}}) \cdot$

$P(w_i = \boxed{\text{again}} \mid i=4, w_{i-4} = \boxed{\text{NONE}}, w_{i-3} = \boxed{\text{The}}, w_{i-2} = \boxed{\text{dog}}, w_{i-1} = \boxed{\text{barked}})$

Language Models – basics

$$P(s) = P(w_1, w_2, \dots w_m) = P(w_1)\, P(w_2|w_1) \dots P(w_m|w_1,\dots,w_{m-1})$$

Position backoff

$P($ The dog barked again $) \approx$

$P(w_i=$ The $|$ $w_{i-1}=$ NONE $) \cdot$

$P(w_i=$ dog $|$ $w_{i-2}=$ NONE $, w_{i-1}=$ The $) \cdot$

$P(w_i=$ barked $|$ $w_{i-3}=$ NONE $, w_{i-2}=$ The $, w_{i-1}=$ dog $) \cdot$

$P(w_i=$ again $|$ $w_{i-4}=$ NONE $, w_{i-3}=$ The $, w_{i-2}=$ dog $, w_{i-1}=$ barked $)$

Language Models – basics

$P(s) = P(w_1, w_2, \ldots w_m) \approx \Pi_{i=1..m} P(w_i \mid w_{i-1})$

History backoff (bigram LM)

P(The dog barked again) ≈

$P(w_i=$ The $\mid$ $w_{i-1}=$ NONE $)\cdot$

$P(w_i=$ dog $\mid$ $w_{i-1}=$ The $)\cdot$

$P(w_i=$ barked $\mid$ $w_{i-1}=$ dog $)\cdot$

$P(w_i=$ again $\mid$ $w_{i-1}=$ barked $)$

Language Models – basics

$P(s) = P(w_1, w_2, \ldots w_m) \approx \Pi_{i=1..m} P(w_i \mid w_{i-2}, w_{i-1})$

History backoff (trigram LM)

P(The dog barked again) ≈

$P(w_i=$ The $\mid w_{i-2}=$ NONE $, w_{i-1}=$ NONE $) \cdot$

$P(w_i=$ dog $\mid w_{i-2}=$ NONE $, w_{i-1}=$ The $) \cdot$

$P(w_i=$ barked $\mid w_{i-2}=$ The $, w_{i-1}=$ dog $) \cdot$

$P(w_i=$ again $\mid w_{i-2}=$ dog $, w_{i-1}=$ barked $)$

Language Models – basics

$P(s) = P(w_1, w_2, ... w_m) \approx \Pi_{i=1..m} P(w_i \mid w_{i-2}, w_{i-1})$

In general: $\Pi_{i=1..m} P(w_i \mid \mathbf{h}_i)$

$\mathbf{h}_i$ … context (history) of word w_i

Language Models – design decisions

1) How to factorize $P(w_1, w_2, \ldots w_m)$ into $\Pi_{i=1..m} P(w_i \mid \mathbf{h}_i)$,
   i.e. what word-positions will be used as the context $\mathbf{h_i}$?

2) What additional context information will be used
   (apart from word forms),
   e.g. stems, lemmata, POS tags, word classes,... ?

3) How to estimate $P(w_i \mid \mathbf{h}_i)$ from the training data?
   Which smoothing technique will be used?
   (Good-Turing, Jelinek-Mercer, Katz, Kneser-Ney,...)
   Generalized Parallel Backoff etc.

Language Models – design decisions

1) How t $P(w_1, w_2, \ldots w_m)$ into $\Pi_{i=1..m} P(w_i \mid \mathbf{h}_i)$,
i.e. wha itions will be used as the context $\mathbf{h}_i$?

this work

2) What add context information will be used
(apart from d forms),
e.g. stems, mmata, POS tags, word classes,... ?

3) How to estimate $P(w_i \mid \mathbf{h}_i)$ from the training data?
que will be used?
ercer, Katz, Kneser-Ney,...)
Generalized Parallel Backoff etc.

Linear interpolation
Weights trained by EM

Language Models – design decisions

1) How t (w$_1$, w$_2$,

i.e. wha itions will

this work

h$_i$ = w$_{i-n+1}$, ..., w$_{i-1}$

(n-gram-based LMs)

2) What add context information will e used

(apart from d forms),

e.g. stems, mmata, POS tags, word ses,... ?

3) How to estimate P(w$_i$ | **h**$_i$) from the tr ata?

que will be

ercer, Katz y,...)

Generalized Parallel Backoff etc.

Linear interpolation

Weights trained by EM

other papers

Outline

- Language Models (LM)
  - basics
  - design decisions
- Post-ngram LM
- Dependency LM
- Evaluation
- Conclusion & future plans

Post-ngram LM

In general: $P(s) = P(w_1, w_2, \ldots w_m) \approx \Pi_{i=1..m} P(w_i \mid \mathbf{h}_i)$

$\mathbf{h}_i$ … context (history) of word w_i

left-to-right factorization order

Bigram LM: $\mathbf{h}_i = w_{i-1}$ (one previous word)

Trigram LM: $\mathbf{h}_i = w_{i-2}, w_{i-1}$ (two previous words)

Post-ngram LM

In general: $P(s) = P(w_1, w_2, \ldots w_m) \approx \Pi_{i=1..m} P(w_i \mid \mathbf{h}_i)$

$\mathbf{h}_i$ … context (history) of word w_i

left-to-right factorization order

Bigram LM: $\mathbf{h}_i = w_{i-1}$ (one previous word)

Trigram LM: $\mathbf{h}_i = w_{i-2}, w_{i-1}$ (two previous words)

right-to-left factorization order

Post-bigram LM: $\mathbf{h}_i = w_{i+1}$ (one following word)

Post-trigram LM: $\mathbf{h}_i = w_{i+1}, w_{i+2}$ (two following words)

Post-ngram LM

In general: $P(s) = P(w_1, w_2, \ldots w_m) \approx \Pi_{i=1..m} P(w_i \mid \mathbf{h}_i)$

$\mathbf{h}_i$ … context (history) of word w_i

left-to-right factorization order

Bigram LM: $\mathbf{h}_i = w_{i-1}$ (one previous word)

Trigram LM: $\mathbf{h}_i = w_{i-2}, w_{i-1}$ (two previous words)

right-to-left factorization order

Post-bigram LM: $\mathbf{h}_i = w_{i+1}$ (one following word)

P(The dog barked again) = P(again | NONE) · P(barked | again) ·
P(dog | barked) · P(The | dog)

Outline

- Language Models (LM)
  - basics
  - design decisions
- Post-ngram LM
- Dependency LM
- Evaluation
- Conclusion & future plans

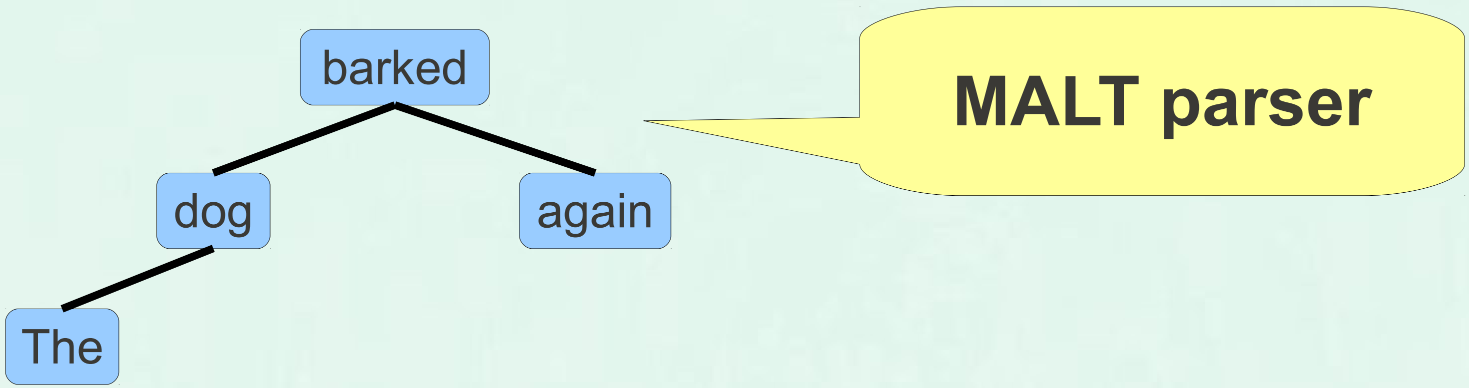
Dependency LM

- exploit the topology of dependency trees

The dog barked again

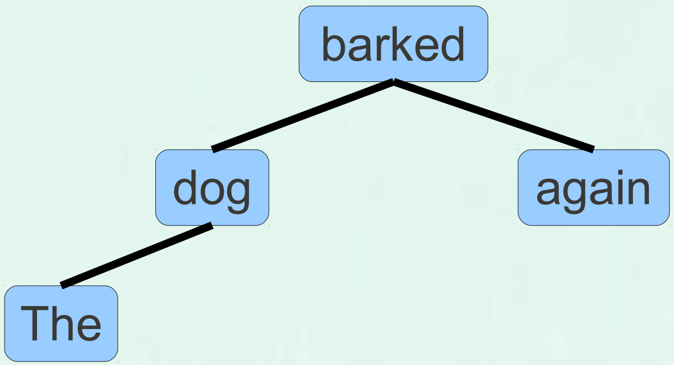
Dependency LM

- exploit the topology of dependency trees



Dependency LM

- exploit the topology of dependency trees



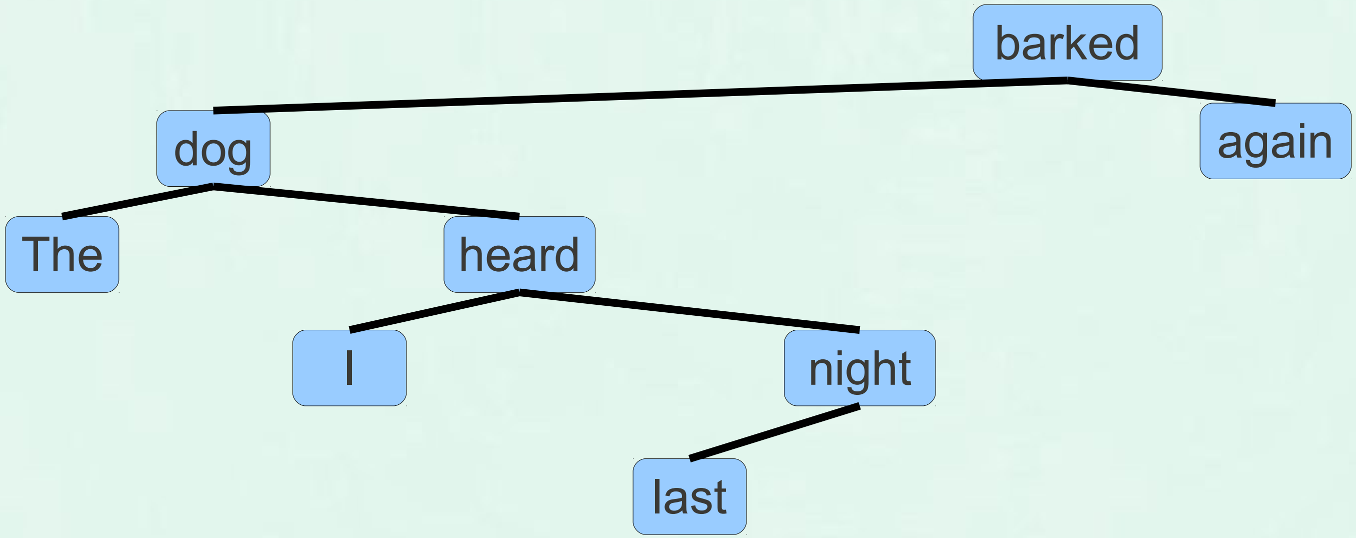
$$P(\text{The dog barked again}) = P(\text{The} \mid \text{dog}) \cdot P(\text{dog} \mid \text{barked}) \cdot P(\text{barked} \mid \text{NONE}) \cdot P(\text{again} \mid \text{barked})$$

$\mathbf{h}_i = \mathrm{parent}(w_i)$

Dependency LM

Long distance dependencies

The dog I heard last night barked again



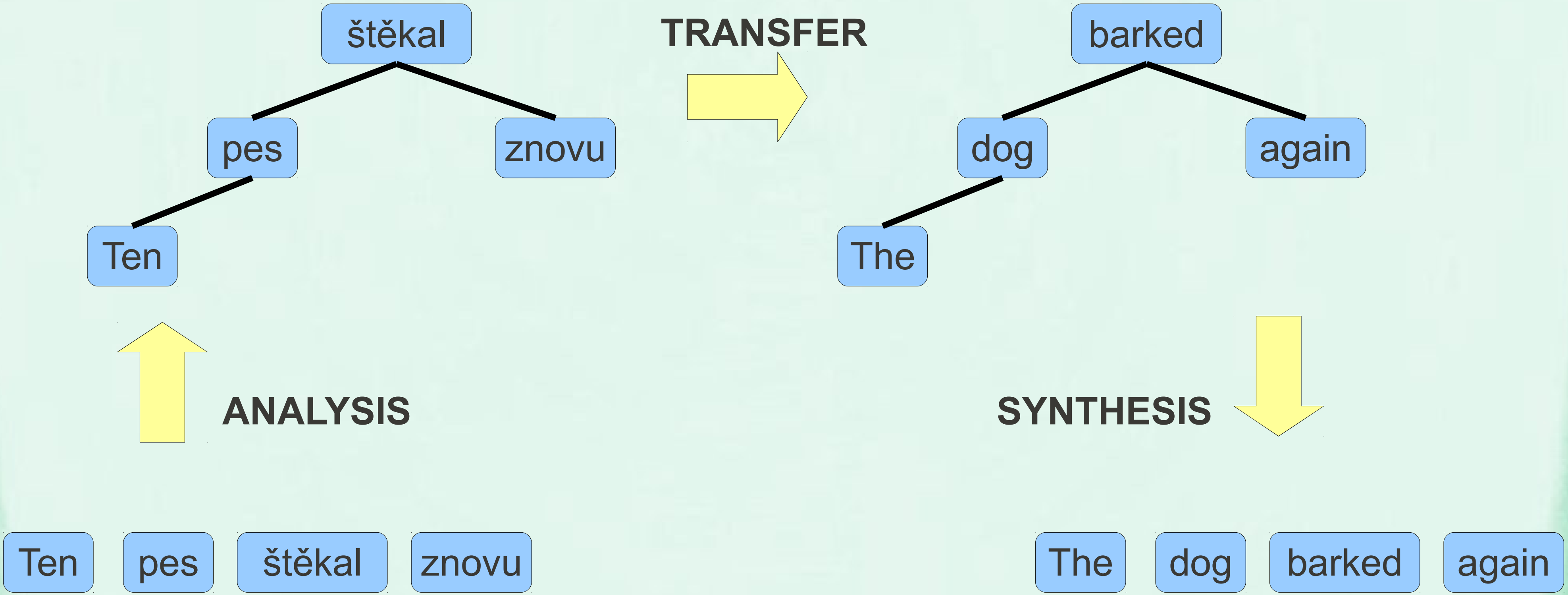
Dependency LM
Motivation for usage

- How can we know the dependency structure without knowing the word-forms?

Dependency LM

Motivation for usage

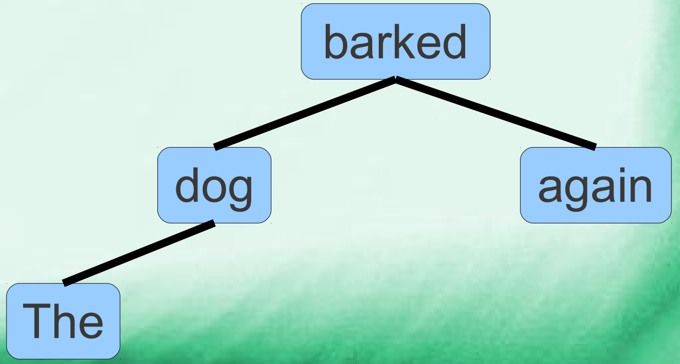
- How can we know the dependency structure without knowing the word-forms?
- For example in tree-to-tree machine translation.



Dependency LM
Examples

- Model **`wp`**
  word form of parent

P(The dog barked again) =
P(The | dog) ·
P(dog | barked) ·
P(barked | NONE) ·
P(again | barked)



Dependency LM
Examples

- Model **`wp,wg`**
  word form of parent, word form of grandparent

P(The dog barked again) =

P(The | dog , barked) ·

P(dog | barked , NONE) ·

P(barked | NONE , NONE) ·

P(again | barked , NONE)

Dependency LM

Examples

- Model **`E,wp`**
  edge direction, word form of parent

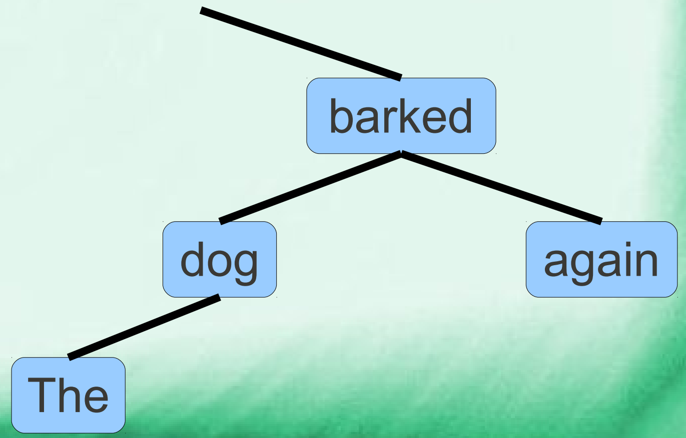
P(The dog barked again) =

P(The | right, dog) ·

P(dog | right, barked) ·

P(barked | left, NONE) ·

P(again | left, barked)



Dependency LM
Examples

- Model **c,wp**
  number of children, word form of parent

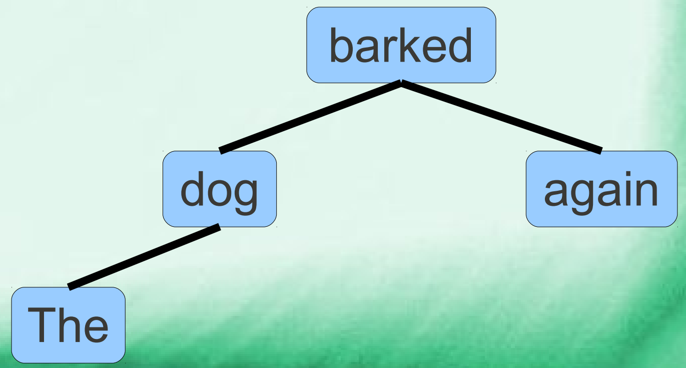
P(The dog barked again) =

P(The | 0, dog) ·

P(dog | 1, barked) ·

P(barked | 2, NONE) ·

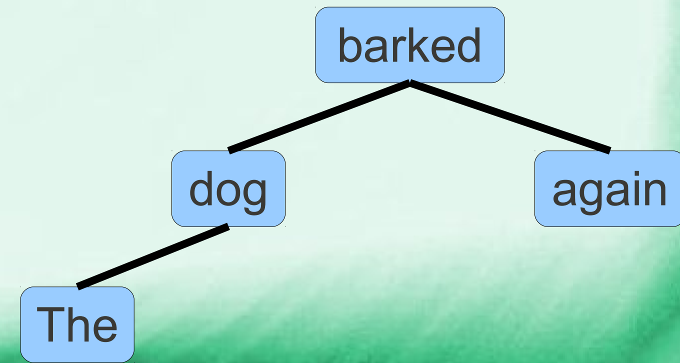
P(again | 0, barked)



Dependency LM
Examples

- Model **`N, wp`**
  the word is Nth child of its parent, word form of parent

P(The dog barked again) =
P(The | 1, dog) ·
P(dog | 1, barked) ·
P(barked | 1, NONE) ·
P(again | 2, barked)



Dependency LM

Examples of additional context information

- Model **`tp,wp`**
  POS tag of parent, word form of parent

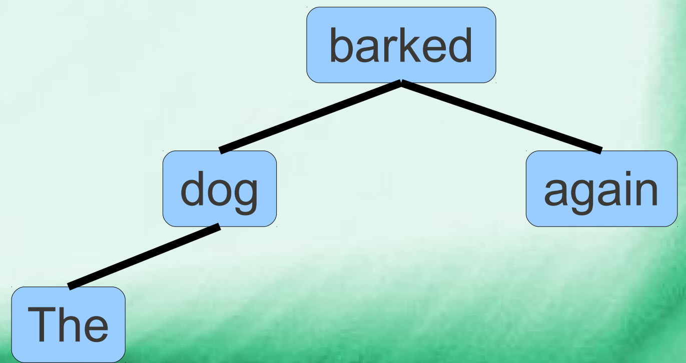
P(The dog barked again) =

P(The | NN, dog) ·

P(dog | VBD, barked) ·

P(barked | NONE, NONE) ·

P(again | VBD, barked)



Dependency LM

Examples of additional context information

- Model **`tp,wp`**

  POS tag of parent, word form of parent

naïve tagger assignes the most frequent tag for a given word

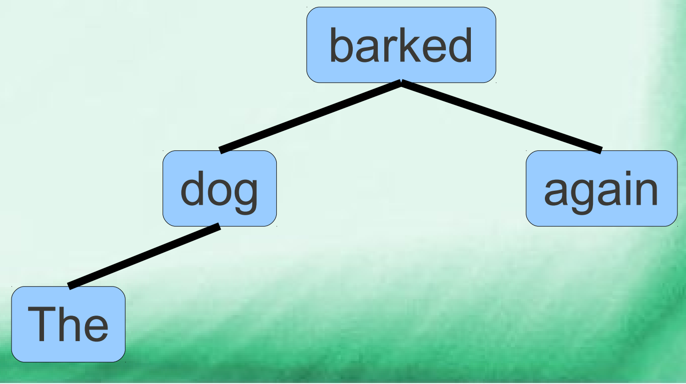
P(The dog barked again) =

P(The | NN, dog) ·

P(dog | VBD, barked) ·

P(barked | NONE, NONE) ·

P(again | VBD, barked)



Dependency LM

Examples of additional context information

- Model **`Tp,wp`**
  coarse-grained POS tag of parent, word form of parent

P(The dog barked again) =

P(The | N, dog) ·

P(dog | V, barked) ·

P(barked | x, NONE) ·

P(again | V, barked)

Dependency LM

Examples of additional context information

- Model **E,C,wp,N**
  edge direction, # children, word form of parent,
  word is N^th child of its parent

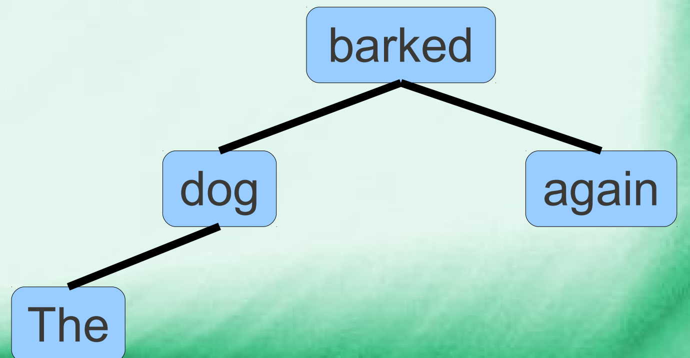
P(The dog barked again) =

P(The | right, 0, dog , 1) ·

P(dog | right, 1, barked , 1) ·

P(barked | left, 2, NONE , 1) ·

P(again | left, 0, barked , 2)



Outline

- Language Models (LM)
  - basics
  - design decisions
- Post-ngram LM
- Dependency LM
- Evaluation
- Conclusion & future plans

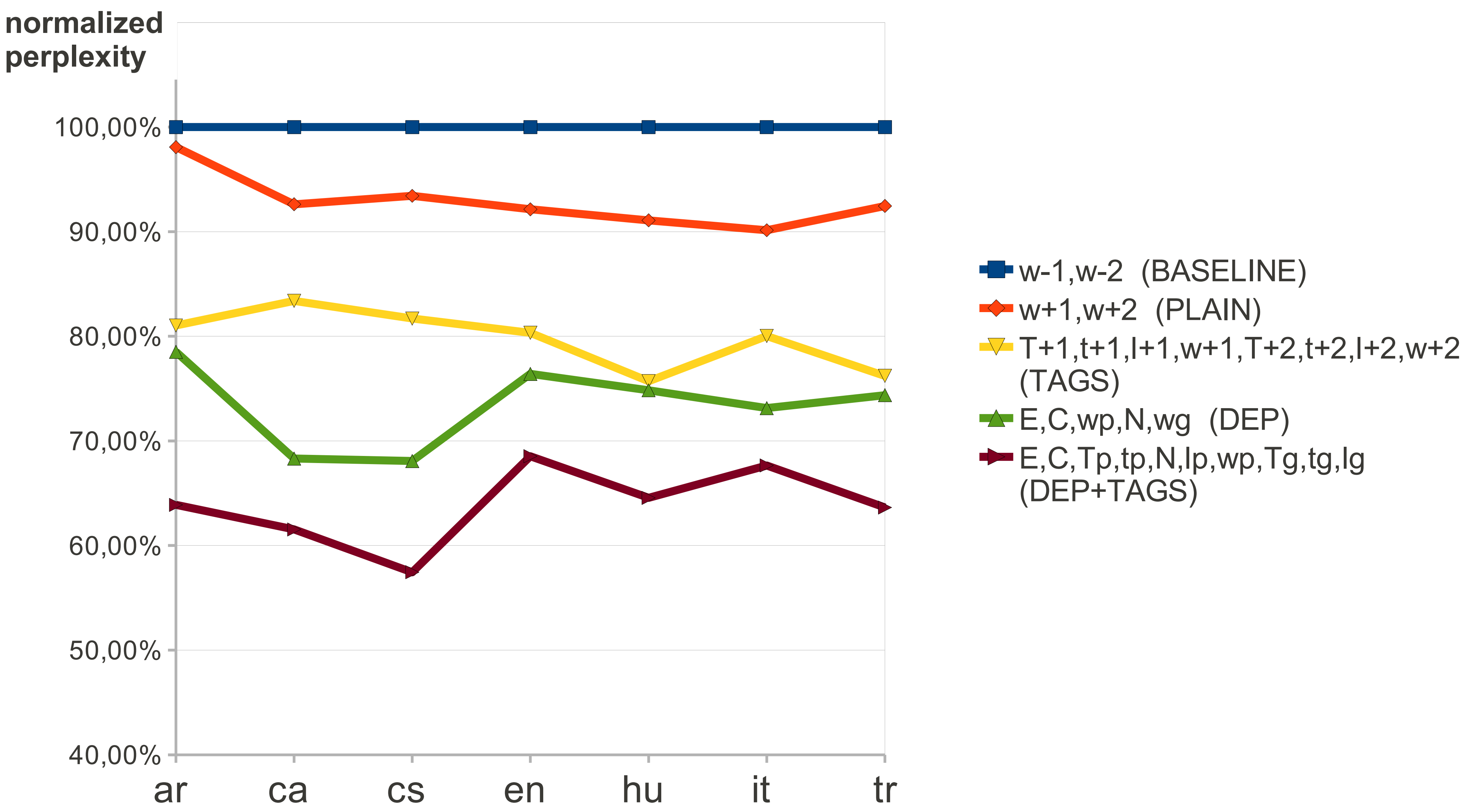
Evaluation

- Train and test data from CoNLL 2007 shared task
- 7 languages: Arabic, Catalan, Czech, English (450 000 tokens, 3 % OOV), Hungarian, Italian (75 000 tokens), and Turkish (26 % OOV)

- Cross-entropy = $-(1/|T|)\ \Sigma_{i=1..|T|} \log_2 P(w_i \mid \mathbf{h}_i)$, measured on the test data T
- Perplexity = $2^{\text{Cross-entropy}}$
- Lower perplexity ~ better LM

- Baseline … trigram LM
- 4 experimental settings: PLAIN, TAGS, DEP, DEP+TAGS

Evaluation



normalized perplexity
100,00%
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
ar
ca
cs
en
hu
it
tr
w-1,w-2 (BASELINE)
w+1,w+2 (PLAIN)
T+1,t+1,l+1,w+1,T+2,t+2,l+2,w+2 (TAGS)
E,C,wp,N,wg (DEP)
E,C,Tp,tp,N,lp,wp,Tg,tg,lg (DEP+TAGS)

Conclusion

Findings confirmed for all seven languages	Improvement over baseline for English	
• Post-trigram better than trigram Post-bigram better than bigram	PLAIN	8 %
• Additional context (POS & lemma) helps	TAGS	20 %
• Dependency structure helps even more	DEP	24 %
• The best perplexity achieved with	DEP+TAGS	31 %

Future plans

- Investigate the reason for better post-ngram LM perplexity
- Extrinsic evaluation
  - Post-ngram LM in speech recognition
  - Dependency LM in tree-to-tree machine translation
- Better smoothing using Generalized Parallel Backoff
- Bigger LM for real applications

Thank you