N-GRAM LANGUAGE MODELS

Evaluating Language Models

Unigram vs. Bigram vs. Trigram vs. n-gram

Evaluating Language Models

The more n in n-gram, the better language model?
The more history, the better prediction of future?

Evaluating Language Models

Qualitative → Generate

Unigram:

```
stream = []
while (w != '</s>'):
    w = unigrams_freq.select()
    stream.append(w)
```

Evaluating Language Models

Qualitative → Generate

Bigram:

```
stream = []
while (w != '</s>'):
    w = bigrams_freq[stream[-1]].select()
    stream.append(w)
```

Evaluating Language Models

Qualitative → Generate

Trigram:

```
stream = []
while (w != '</s>'):
    w = trigrams_freq[stream[-2:]].select()
    stream.append(w)
```

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Figure 3.3 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Figure 3.4 Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
	–Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
	–What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
	–This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
	–It cannot be but so.

Figure 3.3 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Figure 3.4 Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

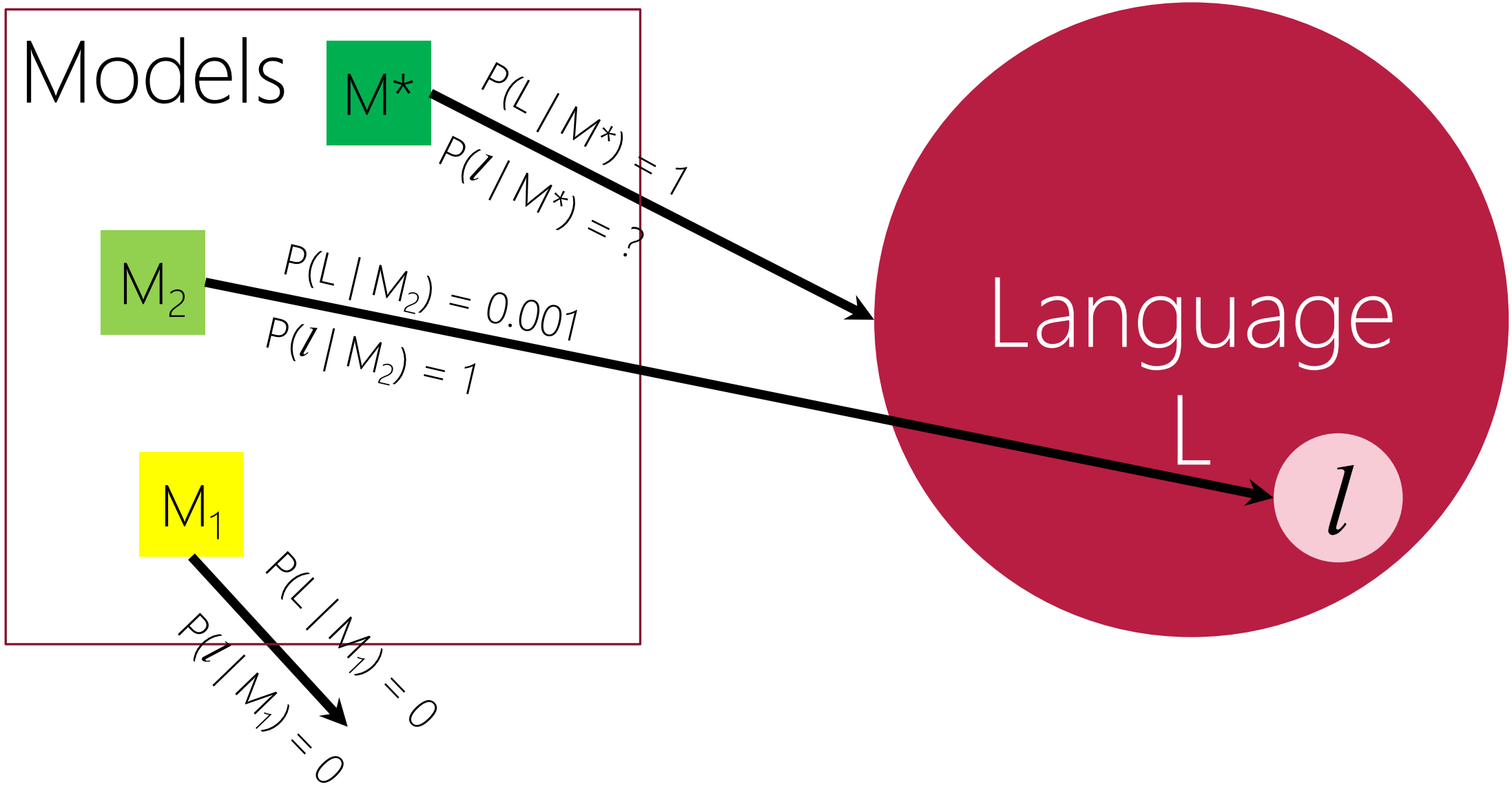
Cross Evaluating Language Models

Qualitative → Generate

Biased toward the corpus! Dialect, Genre, …

Evaluating Language Models

Quantitative → Likelihood


Models
M*
P(L | M*) = 1
P(l | M*) = ?
M2
P(L | M2) = 0.001
P(l | M2) = 1
M1
P(L | M1) = 0
P(l | M1) = 0
Language
L
l

Find M*

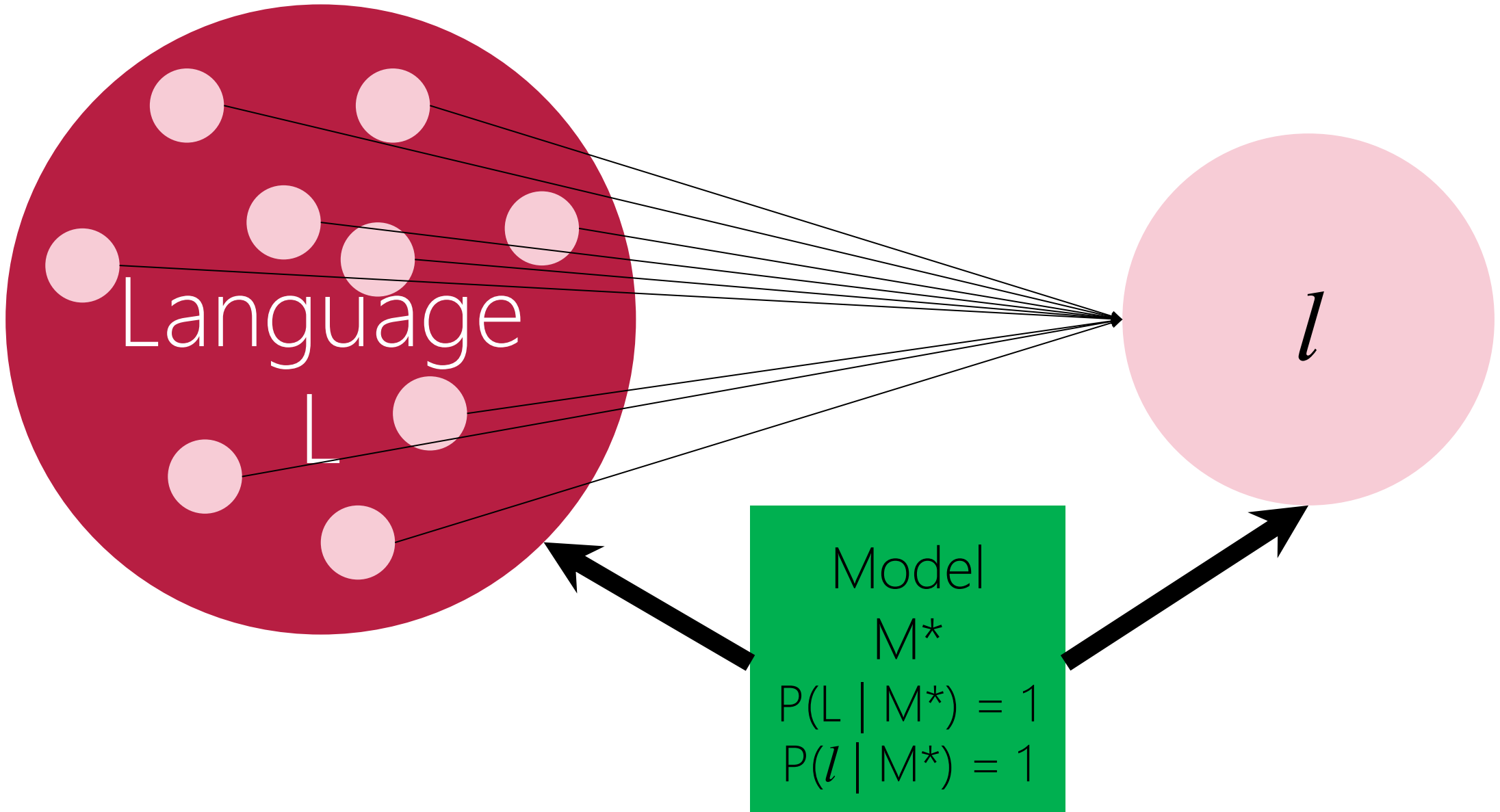
Golden Model

- The language space is available.
- Enumerate the model space to find M*, assuming it exists!

Find M*

Golden Model

- ~~The language space is available.~~ → Randomly selected subset
- Enumerate the model space to find M*, assuming it exists!

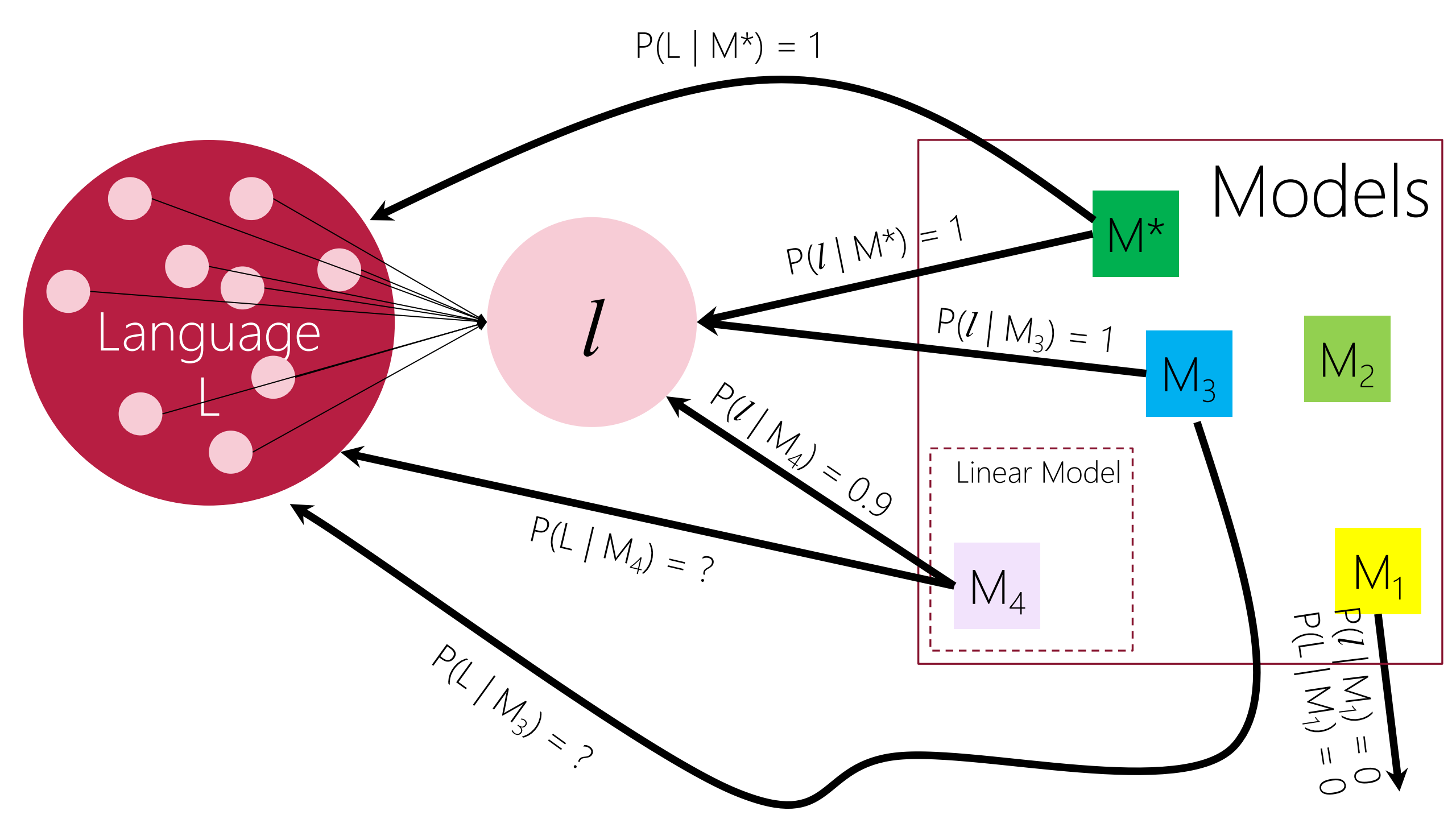

Language
L
l
Model
M*
P(L | M*) = 1
P(l | M*) = 1

Find ~~M*~~ M^

~~Golden~~ Silver Model

- ~~The language space is available.~~ → Randomly selected subset
- ~~Enumerate the model space to find M*~~, assuming it exists!

→ Search subspace, e.g., just linear models!


P(L | M*) = 1
Language
L
l
P(l | M*) = 1
Models
M*
P(l | M3) = 1
M3
M2
P(l | M4) = 0.9
Linear Model
P(L | M4) = ?
M4
M1
P(l | M1) = 0
P(L | M1) = 0
P(L | M3) = ?

$$M\hat{} = \operatorname{argmax}_{M \in \text{Models}} P(l \mid M)$$

- ~~The language space is available.~~ → Randomly selected subset
- ~~Enumerate the model space to find M*~~, assuming it exists!

→ Search subspace, e.g., just linear models!

$$\hat{M} = \operatorname{argmax}_{M \in \text{Models}} P(l \mid M)$$

$P(l \mid M)$ is also called Likelihood → $\mathcal{L}(l \mid M)$
Argmax is also called Maximum Likelihood Estimation → MLE
Argmax sometimes is expressed in log → Log Likelihood

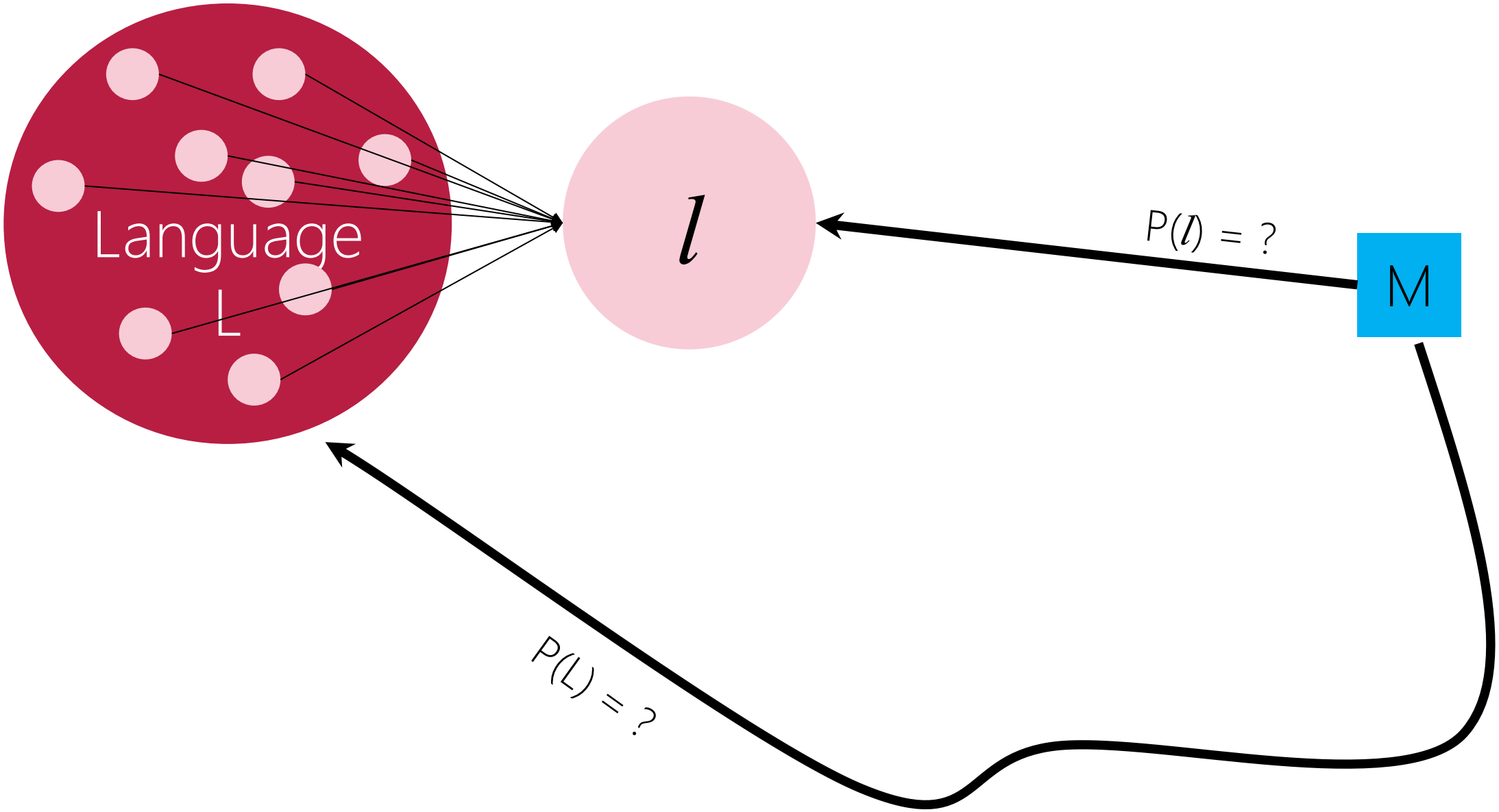
$$P(\boldsymbol{l} \mid M) = \mathcal{L}(\boldsymbol{l} \mid M) = \frac{P(\boldsymbol{l}, M)}{P(M)}$$

P(M) is necessary for models that has prior distribution over their parameters such as Gaussian models $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

$$M\hat{} = \operatorname{argmax}_{M \in \text{Models}} P(l, M)$$

Assuming M is in effect, the likelihood of M is:

$$P(l, M) = P_M(l) = \mathcal{L}_M(l)$$


Language
L
l
P(l) = ?
M
P(L) = ?

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

M1 = |token|-gram model = 22-gram model → $P(l) = \mathcal{L}_{22\text{-gram}}(l) = 1$

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

M1 = |token|-gram model = 22-gram model → $P(l) = \mathcal{L}_{22\text{-gram}}(l) = 1$

M2 = |vocab|-gram model = 16-gram model → $P(l) = \mathcal{L}_{16\text{-gram}}(l)$

P([the])P([course]|[The])P([COMP8730]|[The][course]) …

P([class]|[The][course]…[.][The]..[students][in][the])P(…)

Only the last 15 words

These are not considered in 16-gram model

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

M1 = |token|-gram model = 22-gram model → $P(l) = \mathcal{L}_{22\text{-gram}}(l) = 1$

M2 = |vocab|-gram model = 16-gram model → $P(l) = \mathcal{L}_{16\text{-gram}}(l) = ?$

M3 = 3-gram model = $P(l) = \mathcal{L}_{3\text{-gram}}(l)$

P([The])P([course]|[The])P([COMP8730]|[The][course])P([is]|[course][COMP8730])P([about]|[COMP8730][is]) …

Only the last 2 words

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

M1 = |token|-gram model = 22-gram model → $P(l) = \mathcal{L}_{22\text{-gram}}(l) = 1$

M2 = |vocab|-gram model = 16-gram model → $P(l) = \mathcal{L}_{16\text{-gram}}(l) = ?$

M3 = 3-gram model = $P(l) = \mathcal{L}_{3\text{-gram}}(l) = ?$

M4 = 2-gram model = $P(l) = \mathcal{L}_{2\text{-gram}}(l)$

P([The])P([course]|[The])P([COMP8730]|[course])P([is]|[COMP8730])P([about]|[is])P([nlp]|[about])...

Only the last word → Markovian

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

M1 = |token|-gram model = 22-gram model → P(l) = $\mathcal{L}_{22\text{-gram}}(l)$ = 1
M2 = |vocab|-gram model = 16-gram model → P(l) = $\mathcal{L}_{16\text{-gram}}(l)$ = ?
M3 = 3-gram model = P(l) = $\mathcal{L}_{3\text{-gram}}(l)$ = ?
M4 = 2-gram model = P(l) = $\mathcal{L}_{2\text{-gram}}(l)$ = ?
M5 = 1-gram model = P(l) = $\mathcal{L}_{1\text{-gram}}(l)$
P([The])P([course])P([COMP8730])P([is])P([about])P([nlp])...

No history!

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

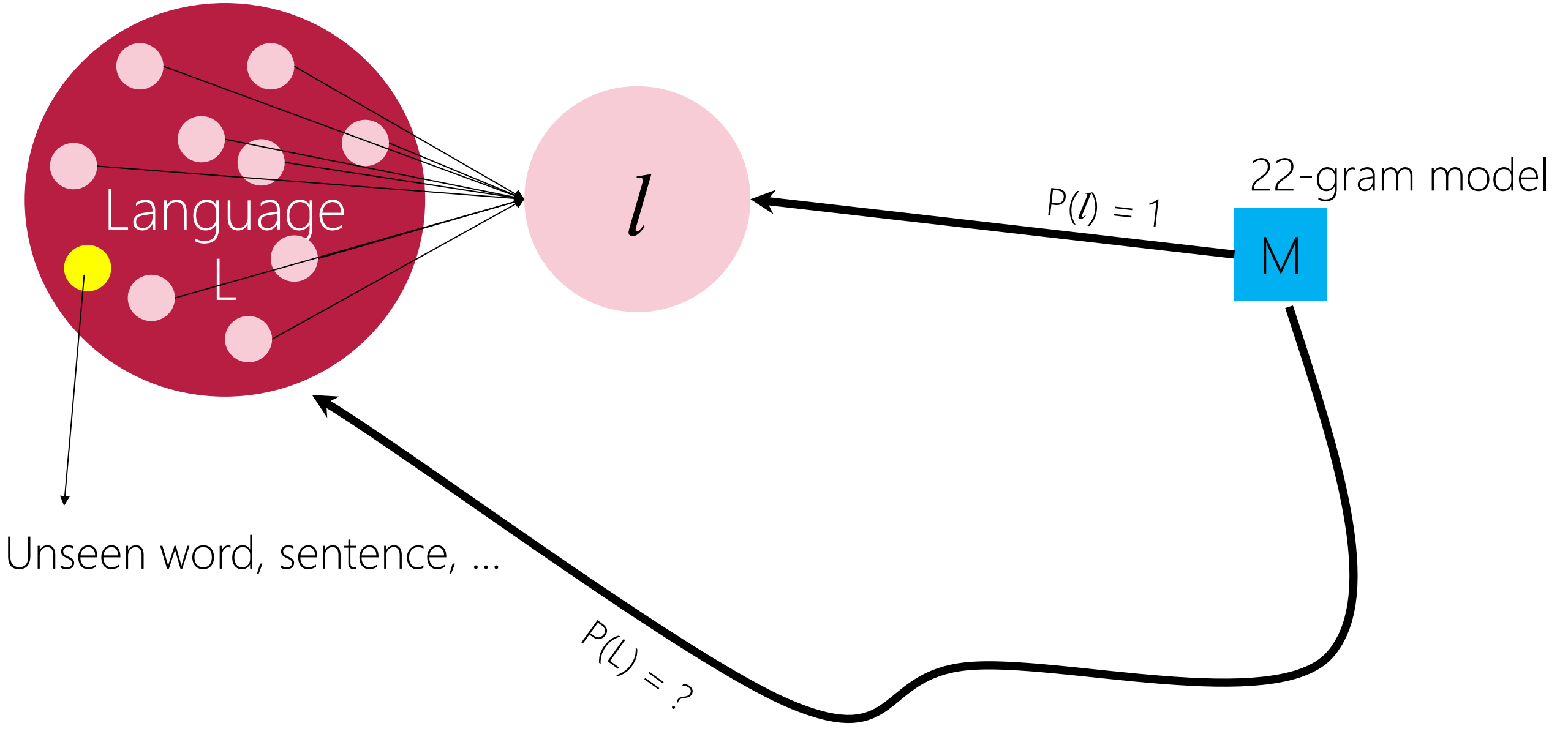
M1 = |token|-gram model = 22-gram model → $P(l) = \mathcal{L}_{22\text{-gram}}(l) = 1$

M2 = |vocab|-gram model = 16-gram model → $P(l) = \mathcal{L}_{16\text{-gram}}(l) = ?$

M3 = 3-gram model = $P(l) = \mathcal{L}_{3\text{-gram}}(l) = ?$

M4 = 2-gram model = $P(l) = \mathcal{L}_{2\text{-gram}}(l) = ?$

M5 = 1-gram model = $P(l) = \mathcal{L}_{1\text{-gram}}(l) = ?$


Language
L
l
P(l) = 1
22-gram model
M
Unseen word, sentence, …
P(L) = ?

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

l': ['Hossein', 'is', 'the', 'name', '.']

M1 = 22-gram model → $P(l') = \mathcal{L}_{22\text{-gram}}(l') = ?$

M2 = 16-gram model → $P(l') = \mathcal{L}_{16\text{-gram}}(l') = ?$

M3 = 3-gram model = $P(l') = \mathcal{L}_{3\text{-gram}}(l') = ?$

M4 = 2-gram model = $P(l') = \mathcal{L}_{2\text{-gram}}(l') = ?$

M5 = 1-gram model = $P(l') = \mathcal{L}_{1\text{-gram}}(l') = ?$

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

l': ['Hossein', 'is', 'the', 'name', '.']

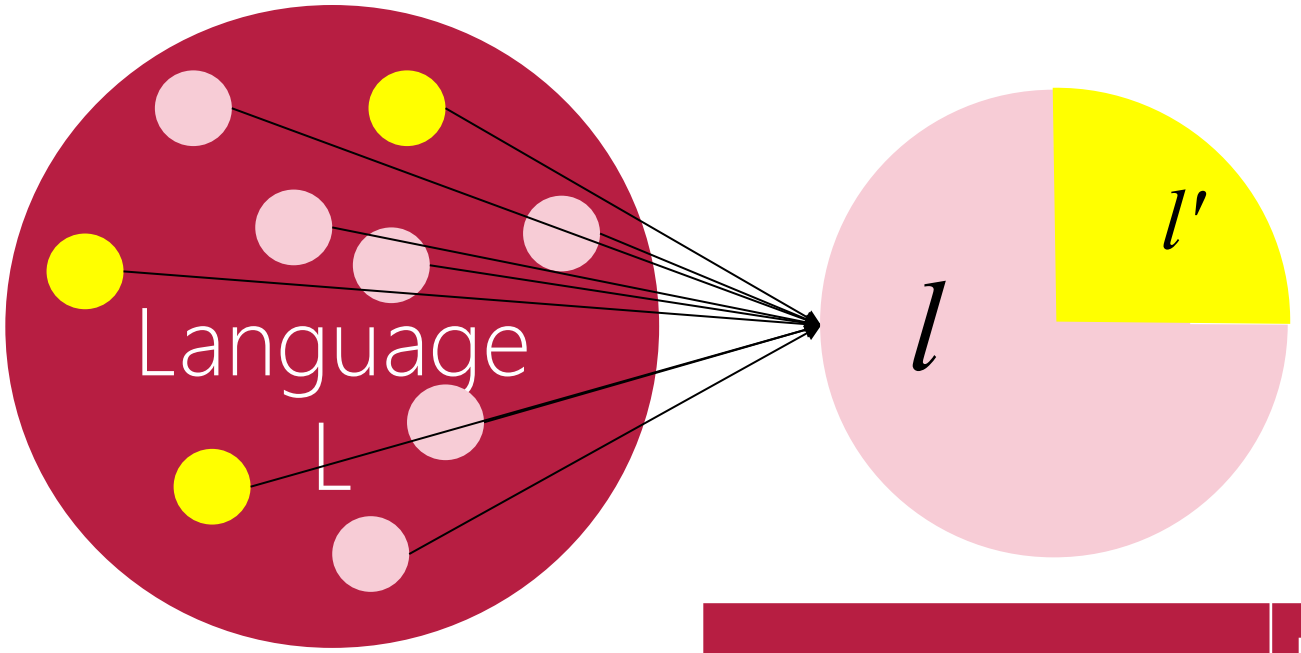
M1 = 22-gram model → $P(l') = \mathcal{L}_{22\text{-gram}}(l') = 0$

M2 = 16-gram model → $P(l') = \mathcal{L}_{16\text{-gram}}(l') = 0$

M3 = 3-gram model = $P(l') = \mathcal{L}_{3\text{-gram}}(l') = 0$

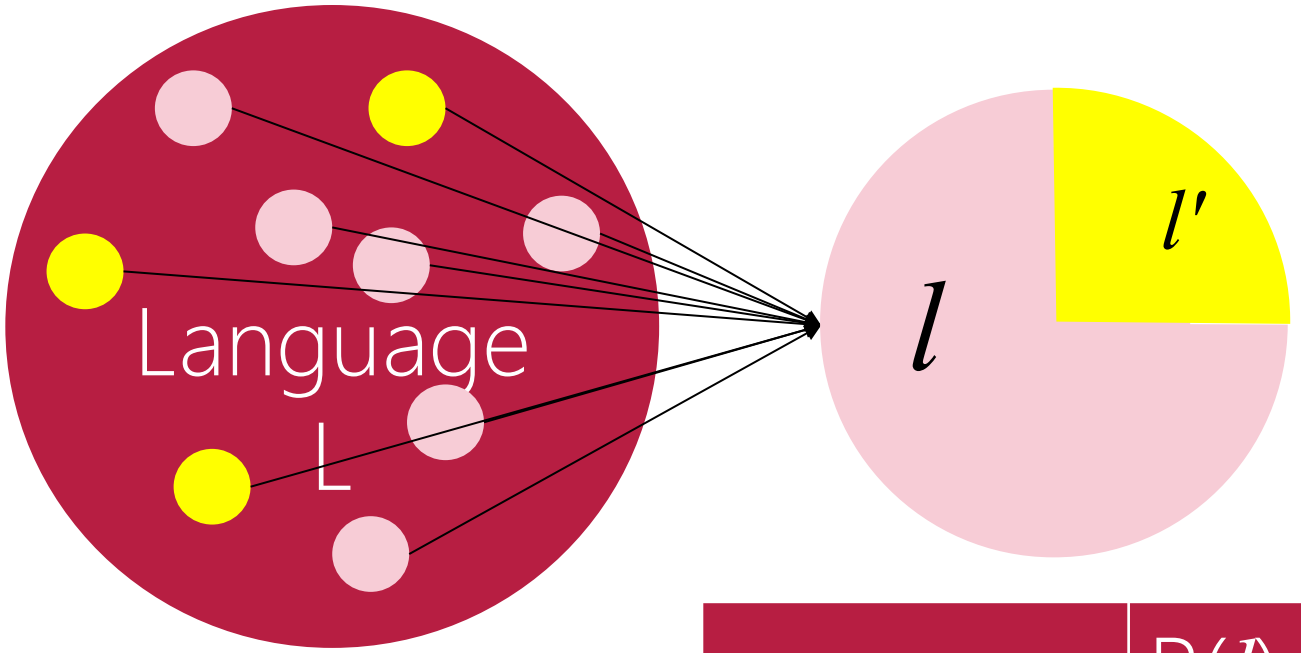
M4 = 2-gram model = $P(l') = \mathcal{L}_{2\text{-gram}}(l') = 0$

M5 = 1-gram model = $P(l') = \mathcal{L}_{1\text{-gram}}(l')$ = nonzero!



	$P(l)=\mathcal{L}_M(l)$	$P(l')=\mathcal{L}_M(l')$
M1	High	High
M2	High	Low
M3	Low	High
M4	Low	Low

P is trained on l and does not know anything about l'



	$P(l)=\mathcal{L}_M(l)$	$P(l')=\mathcal{L}_M(l')$	
M1	High	High	
M2	High	Low	Overfit
M3	Low	High	Not Possible
M4	Low	Low	Underfit

Evaluating Language Models

Quantitative → Perplexity

How perplexed (confused) a language model is to generate seen or unseen stream of tokens! lower perplexity, the better!

$$\text{Perplexity} = \text{PP}(l') = \text{P}(l')^{-1/|l'|}$$

$$= \sqrt[|l'|]{\frac{1}{\text{P}(l')}}$$

P is trained on l and does not know anything about l'

Higher P(l'), lower perplexity, the better!

$$\text{Perplexity} = \text{PP}(l') = \sqrt[|l'|]{\frac{1}{\text{P}(l')}}$$

Unigram approx.: $\sqrt[|l'|]{\frac{1}{\text{P}(l')}} = \sqrt[|l'|]{\frac{1}{\prod_{k=1}^{|l'|} \text{P}(w_k)}}$; $w_i \in l'$

Bigram approx.: $\sqrt[|l'|]{\frac{1}{\text{P}(l')}} = \sqrt[|l'|]{\frac{1}{\prod_{k=1}^{|l'|} \text{P}(w_i|w_{i-1})}}$; $w_{i-1}w_i \in l'$

Trigram approx.: $\sqrt[|l'|]{\frac{1}{\text{P}(l')}} = \sqrt[|l'|]{\frac{1}{\prod_{k=1}^{|l'|} \text{P}(w_i|w_{i-2}w_{i-1})}}$; $w_{i-2}w_{i-1}w_i \in l'$

Corpus: Wall Street Journal
Size: 38 million words
Vocab (Types): 19,979

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Test Set: 1.5 million words

Zeros!

Unseen

Likelihood of a Language Modeling

l: ['The', 'course', 'COMP8730', 'is', 'about', 'nlp', '.', 'The', 'instructor', "'s", name, 'is', 'Hossein', '.', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

l': ['Hossein', 'is', 'the', 'name', 'of', 'a', 'person', '.']

M1 = 22-gram model → $P(l') = \mathcal{L}_{22\text{-gram}}(l') = 0$

M2 = 16-gram model → $P(l') = \mathcal{L}_{16\text{-gram}}(l') = 0$

M3 = 3-gram model = $P(l') = \mathcal{L}_{3\text{-gram}}(l') = 0$

M4 = 2-gram model = $P(l') = \mathcal{L}_{2\text{-gram}}(l') = 0$

M5 = 1-gram model = $P(l') = \mathcal{L}_{1\text{-gram}}(l') =$ 0 (Why?)

Zeros!

Unseen

Not all unigrams are available in training set! E.g., [of]
Not all bigrams are available in training set! E.g., [Hossein][is]
Not all trigrams are available in training set! …

Zeros!

1) Vocabulary + <UNK>
2) Train Vocabulary + Learn Unseen Tokens (Subwords)
3) Smoothing

<UNK>

learn the stats of unseen tokens

1) Pick a dictionary D
2) From $w \in l$ such that $w \notin D$, (OOV) replace it with <UNK>
3) Train model
4) At test, from $w \in l'$, if $w \notin l$ (unseen), replace it with <UNK>

Likelihood of a Language Modeling

D: ['The', 'course', 'is', 'about', 'instructor', 'name', 'There', 'are', '13', 'students', 'in', 'class']

l: ['The', 'course', '<UNK>', 'is', 'about', '<UNK>', '<UNK>', 'The', 'instructor', '<UNK>', name, 'is', '<UNK>', '<UNK>', 'There', 'are', '13', 'students', 'in', 'the', 'class', '.']

l': ['<UNK>', 'is', 'the', 'name', '<UNK>', '<UNK>', '<UNK>', '<UNK>']

M1 = 22-gram model → $P(l') = \mathcal{L}_{22\text{-gram}}(l') = 0$

M2 = 16-gram model → $P(l') = \mathcal{L}_{16\text{-gram}}(l') = 0$

M3 = 3-gram model = $P(l') = \mathcal{L}_{3\text{-gram}}(l') = 0$

M4 = 2-gram model = $P(l') = \mathcal{L}_{2\text{-gram}}(l') = 0$

M5 = 1-gram model = $P(l') = \mathcal{L}_{1\text{-gram}}(l')$ = Nonzero! (Why?)

<UNK>

learn the stats of unseen tokens

1) Pick a dictionary $\boldsymbol{D} \rightarrow$ *small*

Gives higher probability (lower perplexity) at test

All the model should generate is stream of <UNK>s!

Likelihood of a Language Modeling

D: ['The', 'is']

l: ['The', '<UNK>', '<UNK>', 'is', '<UNK>', '<UNK>', '<UNK>', 'The', '<UNK>', '<UNK>', '<UNK>', 'is', '<UNK>', '<UNK>', '<UNK>', '<UNK>', '<UNK>', '<UNK>', '<UNK>', 'the', '<UNK>', '<UNK>']

l': ['<UNK>', 'is', '<UNK>', 'the', '<UNK>', '<UNK>', '<UNK>', '<UNK>', '<UNK>']

M1 = 22-gram model → $P(l') = \mathcal{L}_{22\text{-gram}}(l') = 0$

M2 = 16-gram model → $P(l') = \mathcal{L}_{16\text{-gram}}(l') = 0$

M3 = 3-gram model = $P(l') = \mathcal{L}_{3\text{-gram}}(l') = 0$

M4 = 2-gram model = $P(l') = \mathcal{L}_{2\text{-gram}}(l')$ = Nonzero!

M5 = 1-gram model = $P(l') = \mathcal{L}_{1\text{-gram}}(l')$ = Nonzero!

Subwords

learn the unseen tokens from seen tokens

Please refer to: Text Normalization → Learn to Tokenize

Smoothing

1) Add-1 (Laplace) or Add-k, → k={1,2,…}
2) Backoff
3) Interpolation
4) …

Add-k

Add k unit to all counts, so zero entries become k. Add-1 is also called Laplace.

Unigram model: $P(w_i) = \frac{\#w_i + k}{|\text{tokens}| + k \times |\text{vocabs}|}$

Bigram model: $P(w_i|w_{i-1}) = \frac{\#(w_{i-1}w_i) + k}{\#(w_{i-1}) + k \times ?}$

Trigram model: $P(w_i|w_{i-2}w_{i-1}) = \frac{\#(w_{i-2}w_{i-1}w_i) + k}{\#(w_{i-2}w_{i-1}) + k \times ?}$

Backoff

If n-gram does have not seen, try (n-1)-gram.

Trigram model: $P(w_i|w_{i-2}w_{i-1}) = \frac{\#(w_{i-2}w_{i-1}w_i)}{\#(w_{i-2}w_{i-1})} = 0$

Bigram model: $P(w_i|w_{i-1}) = \frac{\#(w_{i-1}w_i)}{\#(w_{i-1})} = 0$

Unigram model: $P(w_i) = \frac{\#w_i}{|\text{tokens}|}$

Interpolation

P(n-gram) is linear interpolation of all (n-i)-grams: i={1,2,…, n-1}.

Trigram model: $P(w_i|w_{i-2}w_{i-1}) = \lambda_1 \frac{\#(w_{i-2}w_{i-1}w_i)}{\#(w_{i-2}w_{i-1})} +$

Bigram model: $P(w_i|w_{i-1}) = \lambda_2 \frac{\#(w_{i-1}w_i)}{\#(w_{i-1})} +$

Unigram model: $P(w_i) = \lambda_3 \frac{\#w_i}{|\text{tokens}|}$

$$\sum \lambda_i = 1$$

Kneser-Ney Smoothing

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In ICASSP-95, Vol. 1, 181–184.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. Computer Speech and Language, 13, 359–394.