1 Language Models

With a language model, we can estimate how likely a string is English (or how likely that an English speaker would have uttered the string). Naturally, we want an SMT system not only to contain precise translations but also combine them into fluent English.

Language models in general are functions, that take a string and return an estimate of how likely is that string is a proper English phrase.

LMs help us with **reordering**, for example:

$$p_{\text{LM}}(\text{the house is small}) > p_{\text{LM}}(\text{small the is house})$$

Also, they help with the choice of the **right word**:

$$p_{\text{LM}}(\text{I am going home}) > p_{\text{LM}}(\text{I am going house})$$

As a rule, the functions are estimated on large **monolingual** (much more abundant than parallel) corpora of **target** text.

$$\begin{aligned} p(w_1^k) &= p(w_1, w_2, \ldots, w_k) \\ &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \ldots p(w_k|w_1, \ldots w_{k-1}) \end{aligned}$$

It is natural to further assume, that only the previous history of $n-1$ words (for some n) matters for predicting the next word. We are going to use maximum likelihood estimation, so another reason for limiting history is that sufficiently long phrases don't appear in any fixed-size corpus, therefore we must break the computation of the language model into smaller steps.

$$\begin{aligned} p(w_1^k) &= p(w_1, w_2, \ldots, w_k) \\ &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \ldots p(w_i|w_{i-n+1}, \ldots w_{i-1}) \ldots p(w_k|w_{k-n+1}, \ldots w_{k-1}) \end{aligned}$$

This is an application of the chain rule for conditional probabilities:

$$\begin{aligned} p(x|y) &= \frac{p(x, y)}{p(y)} \\ p(y)p(x|y) &= p(x, y) \end{aligned}$$

1.1 n-gram Language Models

1.1.1 Definition of n-gram

The Markov assumption tells us, that only $n-1$ words in history matter.

$$p(w_1,\dots.w_k) \simeq \prod_{i=1}^{k} p(w_i|w_{i-n+1}^{i-1}) \tag{1}$$

bigram: $n=2 \Rightarrow$ **First order Markov model**
Example: $p(w_1,\dots,w_k) \simeq p(w_1)p(w_2|w_1)\dots p(w_k|w_{k-1})$

trigram: $n=3 \Rightarrow$ **Second order Markov model**
Example: $p(w_1,\dots,w_k) \simeq p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)\dots p(w_k|w_{k-2},w_{k-1})$

1.2 Estimating n-gram probabilities

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{\text{count}(w_{i-n+1}^{i})}{\text{count}(w_{i-n+1}^{i-1})}$$

Maximum likelihood estimation:

$$p(w_2|w_1) = \frac{\text{count}(w_1,w_2)}{\text{count}(w_1)} \tag{2}$$

$$p(w_k|w_{k-1},w_{k-2}) = \frac{\text{count}(w_{k-2},w_{k-1},w_k)}{\text{count}(w_{k-2},w_{k-1})}$$

Example from a 3-gram LM trained on the European Parliament proceedings:

the green (total: 1748)		
Word	Count	Prob.
paper	801	0.458
group	640	0.367
light	110	0.063
party	27	0.015
ecu	21	0.012

the red (total: 225)		
Word	Count	Prob.
cross	123	0.547
tape	31	0.138
army	9	0.040
card	7	0.031
,	5	0.022

the blue (total: 54)		
Word	Count	Prob.
box	16	0.296
.	6	0.111
flag	6	0.111
,	3	0.056
angel	3	0.056

1.3 Fit quality: Perplexity

Problem: How well does a language model perform/fit the data? (e.g., what order is enough? which smoothing technique is better?)

One possible measure is Perplexity, which is defined based on cross-entropy:
Entropy & perplexity:

$$
\begin{aligned}
H(p(x)) &= -\mathbb{E}_{p(x)} \log_2 p(x) \\
&= -\sum_x p(x) \log_2 p(x) \\
PP(p(x)) &= 2^{H(p(x))}
\end{aligned}
$$

Now, **entropy of a language** is defined as a limit of per-word entropy:

$$H(L, p(x)) = -\lim_{n\to\infty} \frac{1}{n} \sum_{\forall w_1^n} p(w_1^n) \log p(w_1^n),$$

where w_1^n is a n-word sequence from the language L.

Shannon-McMillan-Breimann Theorem:

$$H(L, p(x)) \simeq -\frac{1}{n} \log p(w_1^n), \text{for large } n$$

Thus, a sufficiently long single sequence is representative for the whole language. We can now compute the perplexity of model using a (large) corpus.

In our case the independent events are the observed n-grams, w_{i-n+1}^i, from sentence s. Therefore, the expectation is taken over the emprical distribution:

$$
\begin{aligned}
H(p_{LM}(s)) &= -\frac{1}{|s| - n + 1} \log_2 p_{LM}(s) \\
&= -\frac{1}{|s| - n + 1} \sum_i^{|s|-n+1} \log_2 p(w_i | w_{i-n+1}^{i-1}). \\
PP &= 2^{H(p_{LM}(s))}
\end{aligned}
$$

The idea is that a model with smaller Perplexity on unseen data is better (model is "less surprised" to see this new data).

Example (for the sentence *"I would like to commend the rapporteur on this work"*):

Prediction	p_{LM}	$-\log_2 p_{\mathrm{LM}}$
p_{LM}(*i*|</s><s>)	0.109	3.197
p_{LM}(would|<s>*i*)	0.144	2.791
p_{LM}(like|i would)	0.489	1.031
p_{LM}(to|would like)	0.905	0.144
p_{LM}(commend|like to)	0.002	8.794
p_{LM}(the|to commend)	0.472	1.084
p_{LM}(rapporteur|commend the)	0.147	2.763
p_{LM}(on|the rapporteur)	0.056	4.150
p_{LM}(his|rapporteur on)	0.194	2.367
p_{LM}(work|on his)	0.089	3.498
p_{LM}(.|his work)	0.290	1.785
p_{LM}(</s>|work .)	0.99999	0.000014
	Average	2.634

Example (comparison of different n-gram models):

Word	Unigram	Bigram	Trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8.342	2.884	2.791	2.791
like	9.129	2.026	1.031	1.290
to	5.081	0.402	0.144	0.113
commend	15.487	12.335	8.794	8.633
the	3.885	1.402	1.084	0.880
rapporteur	10.840	7.319	2.763	2.350
on	6.765	4.140	4.150	1.862
his	10.678	7.316	2.367	1.978
work	9.993	4.816	3.498	2.394
.	4.896	3.020	1.785	1.510
</s>	4.828	0.005	0.000	0.000
Average	8.051	4.072	2.634	2.251
Perplexity	265.136	16.817	6.206	4.758

1.4 Smoothing

High-order n-grams are not very frequent in corpora. Zero probabilities of n-grams destroy our computation of sentence probabilities, since it will be floored to zero.
Smoothing attempts to fill in missing statistics in order to avoid zero probabilities (any product of the form (1) is zero if at least one factor is zero).

1.4.1 Laplace smoothing

"Add-one"-smoothing

$$\text{unsmoothed:} \quad p = \frac{c}{n}, \qquad \begin{matrix} c\text{=number of n-gram in corpus} \\ n\text{=count of history} \end{matrix} \tag{3}$$

$$\text{"Add-one"-smoothing:} \quad p = \frac{c+1}{n+v}, \qquad v = \text{size of vocabulary} \tag{4}$$

"Add α"-smoothing

$$p = \frac{c+\alpha}{n+\alpha v}, \qquad \alpha < 1, \alpha \text{ optimized on held-out set} \tag{5}$$

Example: 2-grams in Europarl

Count	Adjusted count		Test count
c	"add 1"	"add α	t_c
0	0.00378	0.00016	0.00016
1	0.00755	0.95725	0.46235
2	0.01133	1.91433	1.39946
3	0.01511	2.87141	2.34307
4	0.01888	3.82850	3.35202
5	0.02266	4.78558	4.35234
6	0.02644	5.74266	5.33762
8	0.03399	7.65683	7.15074
10	0.04155	9.57100	9.11927
20	0.07931	19.14183	18.95948

$\alpha = 0.00017$
t_c = average count of n-gram in test set that occured c times in training corpus

1.4.2 "Deleted estimate"-smoothing

N_r is the count of n-grams with training count r,
T_r is the count in test data of n-grams with training count r.
We also have an estimate $r^* = \frac{T_r}{N_r}$.
Then we switch train and test and combine the results:

$$r^* = \frac{T_r^1 + T_r^2}{N_r^1 + N_r^2} \tag{6}$$

This is similar to 2-fold cross validation.

Count r	Count of counts N_r	Count in held-out T_r	Exp. count $E[r] = T_r/N_r$	Test count t_c
0	7,515,623,434	938,504	0.00012	0.00016
1	753,777	353,383	0.46900	0.46235
2	170,913	239,736	1.40322	1.39946
3	78,614	189,686	2.41381	2.34307
4	46,769	157,485	3.36860	3.35202
5	31,413	134,653	4.28820	4.35234
6	22,520	122,079	5.42301	5.33762
8	13,586	99,668	7.33892	7.15074
10	9,106	85,666	9.41129	9.11927
20	2,797	53,262	19.04992	18.95948

1.4.3 "Good-Turing"-smoothing

Derivation Question: what is the probability of observing n-gram "in the wild" (i.e. in some test test), given that we saw the n-gram with some frequency in the training set?

Assume we have a sample S of size $|S|$ drawn from the true probability n-gram distribution $p(w)$. Define $c(w)$ to be the number of times n-gram w occurs in S. For integer $r \geq 0$ let $S_r = \{w \ : \ c(w) = r\}$. For example, S_0 is the set of n-grams not seen in S.

Define a random variable M_r (depending on S) to be the probability of drawing an n-gram in the set S_r, that is $M_r = \sum_{w \in S_r} p(w)$. For example, M_0 is the "missing mass" – total probability of words not occuring in the sample S.

Had we known M_r, the true probability $P(w|w \in S_r)$ of drawing again *some* word $w \in S_r$ would be

$$\frac{M_r}{|S_r|}, \tag{7}$$

i.e., total mass divided by total number of distinct elements.

Example: *To see the need for a smoothing, imagine we have sampled a large S where each n-gram occurs exactly once (quite unlikely event). The naive way of estimating M_1 would be*

$$\frac{\text{\# of times } w \text{ occurs in } S \times \text{\# of different words we are ok with}}{\text{total size of } S} = \frac{k \times |S_1|}{|S|} = \frac{1 \times |S|}{|S|} = 1.$$

However, for any reasonable distribution $p(w)$ the probability M_1, given such an unlikely sample S, should be close to 0 .

Let us find the expectation of M_r:

$$\begin{aligned}
\mathbb{E}[M_r] &= \sum_w p(w)P[w \in S_r] \\
&= \sum_w p(w)\binom{|S|}{r}p^r(w)(1-p(w))^{|S|-r} \\
&= \sum_w \binom{|S|}{r}p^{r+1}(w)(1-p(w))^{|S|-(r+1)}(1-p(w))\frac{r+1}{r+1} \\
&= \sum_w \binom{|S|}{r+1}p^{r+1}(w)(1-p(w))^{|S|-(r+1)}(1-p(w))\frac{r+1}{|S|-r} \\
&= \sum_w P[w \in S_{r+1}](1-p(w))\frac{r+1}{|S|-r} \\
&= \frac{r+1}{|S|-r}\sum_w P[w \in S_{r+1}] - \frac{r+1}{|S|-r}\sum_w p(w)P[w \in S_{r+1}] \\
&= \frac{r+1}{|S|-r}\sum_w p(w)|S_{r+1}| - \frac{r+1}{|S|-r}\mathbb{E}[M_{r+1}] \\
&= \frac{r+1}{|S|-r}\mathbb{E}[|S_{r+1}|] - \frac{r+1}{|S|-r}\mathbb{E}[M_{r+1}]
\end{aligned}$$

As $0 \leq M_{r+1} \leq 1$, if $k \ll |S|$ the last term is close to zero. Therefore for $k \ll |S|$ an almost unbiased estimate of M_r is

$$\frac{r+1}{|S|-r}\mathbb{E}[|S_{r+1}|] \simeq \frac{r+1}{|S|}|S_{r+1}|,$$

where we assume that means of (large) $|S_{r+1}|$ can be estimated reliably just by $|S_{r+1}|$.

Final formula Using formula (7) we adjust the actual counts r to expected counts r^* with this formula:

$$r^* = (r+1)\frac{N_{r+1}}{N_r} \tag{8}$$

where $N_r = |S_r|$ is the number of n-grams that occur exactly r times in our corpus and N_0 is the total number of n-grams not occuring in the corpus.
In practice, although very well justified, the GT estimate is not alone for n-gram smooothing, because the formula may give very noisy estimates for large r, where the n_{r+1} statistics not reliable or absent ($= 0$), resulting in suboptimal performance.

Count r	Count of counts N_r	Adjusted count r^*	Test count t
0	7,514,941,065	0.00015	0.00016
1	1,132,844	0.46539	0.46235
2	263,611	1.40679	1.39946
3	123,615	2.38767	2.34307
4	73,788	3.33753	3.35202
5	49,254	4.36967	4.35234
6	35,869	5.32928	5.33762
8	21,693	7.43798	7.15074
10	14,880	9.31304	9.11927
20	4,546	19.54487	18.95948

1.5 Back-Off and Interpolation

Back-Off:

In a given corpus, we may never observe a collocation like "Scottish beer drinkers" or "Scottish beer eaters", because they both have a count of 0. Therefore our smoothing methods will assign them the same probability.

A better idea: **back-off to bigrams**, like "beer drinkers" and "beer eaters".

Interpolation:

Higher and lower order n-gram models have different strengths and weaknesses:

- high-order n-grams are sensitive to more context, but have sparse counts
- low-order n-grams consider only very limited context, but have robust counts

Combine them:

$$\begin{aligned} p_I(w_3|w_1, w_2) = \quad & \lambda_1\, p_1(w_3) \\ & + \lambda_2\, p_2(w_3|w_2) \\ & + \lambda_3\, p_3(w_3|w_1, w_2) \end{aligned}$$

With a lot of training data, we can trust the higher order language models more and assign them higher weights. We require that:

$$\begin{aligned} \forall \lambda_n : 0 \leq \lambda_n \leq 1 \\ \sum_n \lambda_n = 1 \end{aligned}$$

Recursive Interpolation (Jelinek-Mercer smoothing)

Recursive definition of interpolation:

$$\begin{aligned} p_n^I(w_i|w_{i-n+1},...,w_{i-1}) = &\lambda_{w_{i-n+1},...,w_{i-1}} \; p_n(w_i|w_{i-n+1},...,w_{i-1}) \\ &+ (1-\lambda_{w_{i-n+1},...,w_{i-1}}) \; p_{n-1}^I(w_i|w_{i-n+2},...,w_{i-1}) \end{aligned}$$

Training $\lambda_{w_{i-n+1},...,w_{i-1}}$ can be done with the EM algorithm, however, partitioning them into buckets according to $c(w_{i-n+1}^{i-1})$ and using the same λ for all counts in the same buckets has a good balance of quality and eficiency.

Recursive Back-Off (Katz smoothing)

To fix the GT smoothing we would trust the highest order language model that contains an n-gram:

$$p_n^{BO}(w_i|w_{i-n+1},...,w_{i-1}) = \begin{cases} d_n(w_{i-n+1},...,w_{i-1}) \; p_n(w_i|w_{i-n+1},...,w_{i-1}) \\ \qquad \text{if } \text{count}_n(w_{i-n+1},...,w_i) > k \\ \alpha_n(w_i|w_{i-n+1},...,w_{i-1}) \; p_{n-1}^{BO}(w_i|w_{i-n+2},...,w_{i-1}) \\ \qquad \text{otherwise} \end{cases}$$

The constant k is usually set to be a small integer, often zero or found emprically by cross-validation.

First note that for GT smoothing the total probability of lost probability mass due to smoothing and that is distributed among unseen n-grams is equal to n_1/N, where $N = \sum_{r>0} r n_r$, with the contribution of each n-gram with count r to be equal to $(1-d_r)\frac{r}{N}$.

Consider two cases:

1. $k = 0$

   - as the GT estimation reduces counts (and estimated probabilites) we can just use the discount values GT smoothing gives as a discounting function $d_n(w_1,...,w_{n-1})$.
   - coefficients $\alpha_n(w_i|w_{i-n+1},...,w_{i-1})$ then just collect all the missing probability mass:

     $$\beta(w_{i-n+1},...,w_{i-1}) \equiv 1 - \sum_{\text{count}_n(w_{i-n+1},...,w_i)>k} d_n(w_{i-n+1},...,w_{i-1}) \; p_n(w_i|w_{i-n+1},...,w_{i-1})$$

     $$\alpha_n(w_i|w_{i-n+1},...,w_{i-1}) = \frac{\beta(w_{i-n+1},...,w_{i-1})}{\sum_{\text{count}_n(w_{i-n+1},...,w_i)\le k} p_{n-1}^{BO}(w_i|w_{i-n+2},...,w_{i-1})}$$

2. $k > 0$

   - set d_r for all n-grams occuring more than k times to be equal to 1 (i.e., considered to be a reliable estimate)
   - otherwise require that contributions d_r of seen n-grams are propotional to the GT contributions

$$1 - d_r = \mu(1 - r^*/r)$$

   - still the total distributed mass as dictated by GT smoothing should be left unchanged

$$\sum_{r>0} n_r(1 - d_r)\frac{r}{N} = \frac{n_1}{N}$$

   - can be shown that the solution is

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}$$

Recursion grounding

- 1-order model: ML (or some smoothed) unigram model
- 0-order model: uniform model $p(w_i) = 1/|V|$