

From Language Modelling to Machine Translation

Phil Blunsom

pblunsom@google.com



UNIVERSITY OF
OXFORD

DEPARTMENT OF
**COMPUTER
SCIENCE**

Language models: the traditional view¹

- **Language models** answer the question:

How likely is it that we observe this string of English words?

- Helps distinguish correct word order:

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

- Helps with word choice:

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

¹LM slides adapted from Philipp Koehn's excellent open source MT course.

Language models: another view

Most of Natural Language Processing can be structured as (conditional) language modeling:

Translation

$$p_{\text{LM}}(\text{Les chiens aiment les os} \parallel \text{Dogs love bones})$$

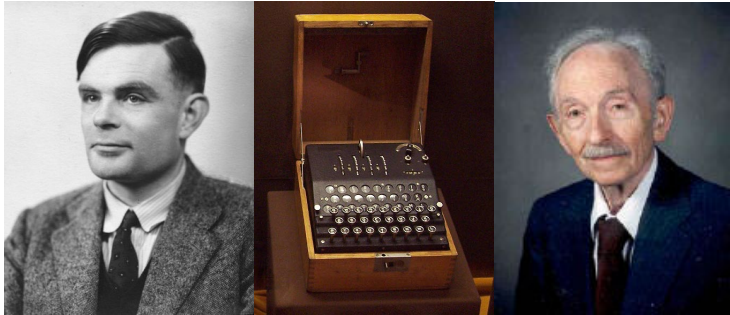
Question Answering

$$p_{\text{LM}}(\text{What do dogs love?} \parallel \text{bones} \cdot \mid \beta)$$

Dialogue

$$p_{\text{LM}}(\text{How are you?} \parallel \text{Fine thanks. And you?} \mid \beta)$$

History: cryptography



N-gram language models

- Given: a string of English words $W = w_1, w_2, w_3, \dots, w_n$
 - Question: what is $p(W)$?
 - Sparse data: Many good English sentences will not have been seen before
- Decomposing $p(W)$ using the chain rule:

$$p(w_1, w_2, w_3, \dots, w_n) = p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) \times \\ \dots \times p(w_n|w_1, w_2, \dots, w_{n-1})$$

(not much gained yet, $p(w_n|w_1, w_2, \dots, w_{n-1})$ is equally sparse)

The Traditional Markov Chain

- **Markov assumption:**
 - only previous history matters
 - limited memory: only last k words are included in history (older words less relevant)
→ **k th order Markov model**
- For instance 2-gram language model:

$$p(w_1, w_2, w_3, \dots, w_n) \simeq p(w_1) p(w_2|w_1) p(w_3|w_2) \dots p(w_n|w_{n-1})$$

- The conditioning context, w_{i-1} , is called the **history**

Estimating N-Gram Probabilities

- Maximum likelihood estimation

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

- Collect counts over a large text corpus
- Millions to billions of words are easy to get
(trillions of English words available on the web)

How good is a LM?

A good model assigns a text of real English W a high probability.
This can be measured with cross entropy:

$$H(W) = -\frac{1}{n} \log_2 p(W_1^n)$$

Intuition 1: Cross entropy is a measure of how many bits are needed to encode text with our model.

Alternatively we can use **perplexity**:

$$\text{perplexity}(W) = 2^{H(W)}$$

Intuition 2: Perplexity is a measure of how surprised our model is on seeing each word.

Comparison 1-4-Gram

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

Unseen N-Grams

- We have seen *i like to* in our corpus
- We have never seen *i like to smooth* in our corpus

$$\rightarrow p(\text{smooth} | \text{i like to}) = 0$$

- Any sentence that includes *i like to smooth* will be assigned probability 0

Add-One Smoothing

- For all possible n-grams, add the count of one.

$$p = \frac{c + 1}{n + v}$$

- c = count of n-gram in corpus
- n = count of history
- v = vocabulary size
- But there are many more unseen n-grams than seen n-grams
- Example: Europarl 2-bigrams:
 - 86,700 distinct words
 - $86,700^2 = 7,516,890,000$ possible bigrams
 - but only about 30,000,000 words (and bigrams) in corpus

Add- α Smoothing

- Add $\alpha < 1$ to each count

$$p = \frac{c + \alpha}{n + \alpha v}$$

- What is a good value for α ?
- Could be optimized on held-out set

Example: 2-Grams in Europarl

Count	Adjusted count		Test count
c	$(c + 1)\frac{n}{n+v^2}$	$(c + \alpha)\frac{n}{n+\alpha v^2}$	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

- Add- α smoothing with $\alpha = 0.00017$
- t_c are average counts of n-grams in test set that occurred c times in corpus

Good-Turing Smoothing

Adjust actual counts r to expected counts r^* with formula

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

N_r number of n-grams that occur exactly r times in corpus

Derivation sketch: estimate the expectation of the probability of a given ngram (α_i) that occurs r times in the corpus:

$$r^* = N \times \mathbb{E}[p_i | \text{count}(\alpha_i) = r].$$

See the references for the complete derivation.

Good-Turing for 2-Grams in Europarl

Count	Count of counts	Adjusted count	Test count
r	N_r	r^*	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

Adjusted count fairly accurate when compared against the test count

Back-Off

- In given corpus, we may never observe
 - Montreal poutine eaters
 - Montreal poutine drinker
- Both have count 0
 - our smoothing methods will assign them the same probability
- Better: backoff to bigrams:
 - poutine eaters
 - poutine drinker

Back-Off

- Trust the highest order language model that contains n-gram

$$p_n^{BO}(w_i | w_{i-n+1}, \dots, w_{i-1}) = \begin{cases} \alpha_n(w_i | w_{i-n+1}, \dots, w_{i-1}) & \text{if } \text{count}_n(w_{i-n+1}, \dots, w_i) > 0 \\ d_n(w_{i-n+1}, \dots, w_{i-1}) p_{n-1}^{BO}(w_i | w_{i-n+2}, \dots, w_{i-1}) & \text{else} \end{cases}$$

- Requires
 - adjusted prediction model $\alpha_n(w_i | w_{i-n+1}, \dots, w_{i-1})$
 - discounting function $d_n(w_1, \dots, w_{n-1})$

Back-Off with Good-Turing Smoothing

- Previously, we computed n-gram probabilities based on relative frequency

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

- Good Turing smoothing adjusts counts c to expected counts c^*

$$\text{count}^*(w_1, w_2) \leq \text{count}(w_1, w_2)$$

- We use these expected counts for the prediction model (but 0^* remains 0)

$$\alpha(w_2|w_1) = \frac{\text{count}^*(w_1, w_2)}{\text{count}(w_1)}$$

- This leaves probability mass for the discounting function

$$d_2(w_1) = 1 - \sum_{w_2} \alpha(w_2|w_1)$$

Diversity of Predicted Words

- Consider the bigram histories **spite** and **constant**
 - both occur 993 times in Europarl corpus
 - only 9 different words follow **spite**
almost always followed by **of** (979 times), due to expression **in spite of**
 - 415 different words follow **constant**
most frequent: **and** (42 times), **concern** (27 times), **pressure** (26 times),
but huge tail of singletons: 268 different words
- More likely to see new bigram that starts with **constant** than **spite**

Witten-Bell smoothing considers diversity of predicted words

Diversity of Histories

- Consider the word **York**:
 - fairly frequent word in Europarl corpus, occurs 477 times,
 - as frequent as **foods**, **indicates** and **providers**,
 - in unigram language model: a respectable probability.
- However, it almost always directly follows **New** (473 times).
- Recall that the unigram model alone is used if the bigram model is inconclusive:
 - **York** unlikely second word in unseen bigram,
 - in back-off unigram model, **York** should have low probability.

Kneser-Ney smoothing takes diversity of histories into account.

Evaluation

Evaluation of ngram smoothing methods:

Perplexity for language models trained on the Europarl corpus

Smoothing method	bigram	trigram	4-gram
Good-Turing	96.2	62.9	59.9
Witten-Bell	97.1	63.8	60.4
Modified Kneser-Ney	95.4	61.6	58.6
Interpolated Modified Kneser-Ney	94.5	59.3	54.0

Provisional Summary

Language models: *How likely is this string of English words?*

- N-gram models (Markov assumption)
- Perplexity
- Count smoothing
 - add-one, add- α
 - Good Turing
- Interpolation and backoff
 - Good Turing
 - Witten-Bell
 - Kneser-Ney

Reference:

An empirical study of smoothing techniques for language modeling.

Stanley Chen and Joshua Goodman. Harvard University, 1998

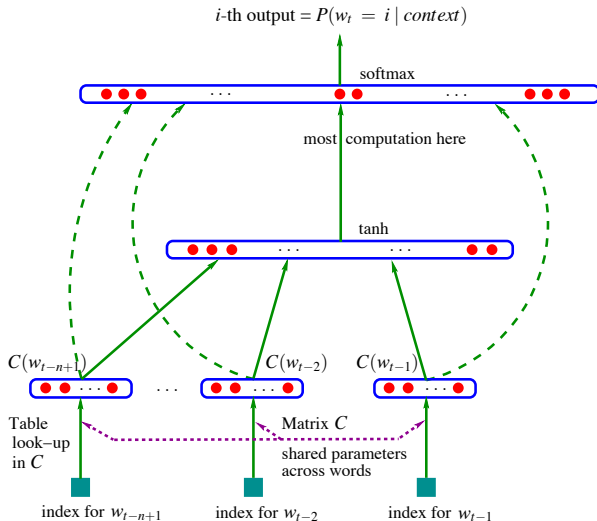
<http://research.microsoft.com/en-us/um/people/joshuago/tr-10-98.pdf>

Provisional Summary

Count based ngram language models are very scalable, but:

- large ngrams are sparse, so hard to capture long range dependencies,
- symbolic nature does not capture correlations between semantically similar word distributions, e.g. cat \leftrightarrow dog,
- similarly morphological regularities, running \leftrightarrow jumping, or gender etc.

Neural language models



A Neural Probabilistic Language Model. Bengio et al. JMLR 2003.

Log-linear models for classification

Features: $\phi(\mathbf{x}) \in \mathbb{R}^D$ and weights: $\lambda_k \in \mathbb{R}^D$ for $k \in \{1, \dots, K\}$ classes:

$$p(C_k|\mathbf{x}) = \frac{\exp(\lambda_k^\top \phi(\mathbf{x}))}{\sum_j^K \exp(\lambda_j^\top \phi(\mathbf{x}))}$$

Gradient required for training:

$$\begin{aligned} \frac{\partial}{\partial \lambda_j} \left[-\log p(C_k|\mathbf{x}) \right] &= \frac{\partial}{\partial \lambda_j} \log \mathcal{Z}(\mathbf{x}) - \frac{\partial}{\partial \lambda_j} \lambda_k^\top \phi(\mathbf{x}) \\ &= \frac{1}{\mathcal{Z}(\mathbf{x})} \frac{\partial}{\partial \lambda_j} \exp(\lambda_j^\top \phi(\mathbf{x})) - \frac{\partial}{\partial \lambda_j} \lambda_k^\top \phi(\mathbf{x}) \\ &= \frac{\exp(\lambda_j^\top \phi(\mathbf{x}))}{\mathcal{Z}(\mathbf{x})} \phi(\mathbf{x}) - \frac{\partial}{\partial \lambda_j} \lambda_k^\top \phi(\mathbf{x}) \\ &= \underbrace{p(C_j|\mathbf{x}) \phi(\mathbf{x})}_{\text{expected features}} - \underbrace{\delta(j, k) \phi(\mathbf{x})}_{\text{observed features}} \end{aligned}$$

$\delta(j, k)$ is the Kronecker delta function which is 1 if $j = k$ and 0 otherwise, and $\mathcal{Z}(\mathbf{x}) = \sum_j^K \exp(\lambda_j^\top \phi(\mathbf{x}))$ is referred to as the partition function.

A simple log-linear (tri-gram) language model

this is America

$$\Phi_1(w_{n-1}, w_{n-2}) = \begin{cases} 1, & \text{if } w_{n-2}=\text{this} \ \& \ w_{n-1}=\text{is} \\ 0, & \text{otherwise} \end{cases}$$

$$\Phi_2(w_{n-1}, w_{n-2}) = \begin{cases} 1, & \text{if } w_{n-2}=\text{this} \\ 0, & \text{otherwise} \end{cases}$$

$$\Phi_3(w_{n-1}, w_{n-2}) = \begin{cases} 1, & \text{if } w_{n-1}=\text{is} \\ 0, & \text{otherwise} \end{cases}$$

Classify the next word w_n given w_{n-1}, w_{n-2} : Features:

$\phi(w_{n-1}, w_{n-2}) \in \mathbb{R}^D$ and weights: $\lambda_i \in \mathbb{R}^D$:²

$$p(w_n | w_{n-1}, w_{n-2}) \propto \exp(\lambda_{w_n}^\top \phi(w_{n-1}, w_{n-2}) + b_{w_n})$$

Traditionally the feature maps $\phi(\cdot)$ are rule based, but can we learn them from the data?

²we now explicitly include a per-word bias parameter b_{w_n} that is initialised to the empirical log $p(w_n)$.

A simple log-linear (tri-gram) language model

this is America

$$\Phi_1(w_{n-1}, w_{n-2}) = \begin{cases} 1, & \text{if } w_{w-2}=\text{this} \ \& \ w_{n-1}=\text{is} \\ 0, & \text{otherwise} \end{cases}$$

$$\Phi_2(w_{n-1}, w_{n-2}) = \begin{cases} 1, & \text{if } w_{w-2}=\text{this} \\ 0, & \text{otherwise} \end{cases}$$

$$\Phi_3(w_{n-1}, w_{n-2}) = \begin{cases} 1, & \text{if } w_{w-1}=\text{is} \\ 0, & \text{otherwise} \end{cases}$$

Traditionally the feature maps $\phi(\cdot)$ are rule based, but can we learn them from the data?

Assume the features factorise across the context words:

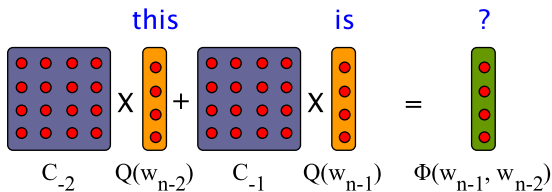
$$p(w_n | w_{n-1}, w_{n-2}) \propto \exp \left(\lambda_{w_n}^T (\phi_{-1}(w_{n-1}) + \phi_{-2}(w_{n-2})) + b_{w_n} \right)$$

Learning the features: the log-bilinear language model

Represent the context words by the columns of a $D \times |\text{vocab}|$ matrix Q , and output words by the columns of a matrix R ; assume ϕ_i is a linear function of these representations parametrised by a matrix C_i :

$$C_{-2} \times Q(w_{n-2}) + C_{-1} \times Q(w_{n-1}) = \Phi(w_{n-1}, w_{n-2})$$

Learning the features: the log-bilinear language model

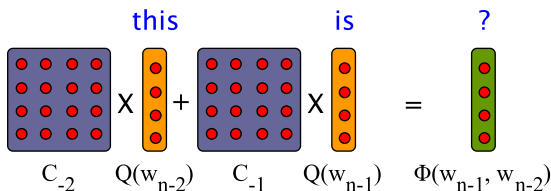


$$\phi(w_{n-1}, w_{n-2}) = C_{-2}Q(w_{n-2}) + C_{-1}Q(w_{n-1})$$
$$p(w_n | w_{n-1}, w_{n-2}) \propto \exp(R(w_n)^T \phi(w_{n-1}, w_{n-2}) + b_{w_n})$$

This is referred to as a *log-bilinear model*.³

³Three new graphical models for statistical language modelling. Mnih and Hinton, ICML'07.

Learning the features: the log-bilinear language model

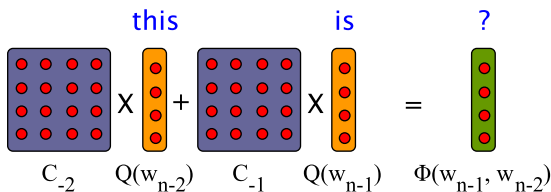


$$p(w_n | w_{n-1}, w_{n-2}) \propto \exp(R(w_n)^T \phi(w_{n-1}, w_{n-2}) + b_{w_n})$$

Error objective: $E = -\log p(w_n | w_{n-1}, w_{n-2})$

$$\begin{aligned} \frac{\partial}{\partial R(j)} E &= \frac{\partial}{\partial R(j)} \log \mathcal{Z}(w_{n-1}, w_{n-2}) - \frac{\partial}{\partial R(j)} R(w_n)^T \phi \\ &= \left(p(j | w_{n-1}, w_{n-2}) - \delta(j, w_n) \right) \phi \end{aligned}$$

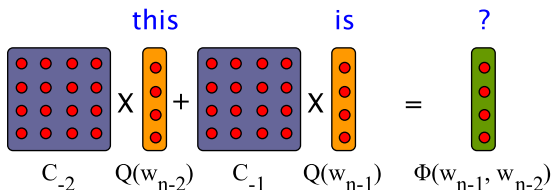
Learning the features: the log-bilinear language model



Error objective: $E = -\log p(w_n | w_{n-1}, w_{n-2})$

$$\begin{aligned} \frac{\partial}{\partial \phi} E &= \frac{\partial}{\partial \phi} \log \mathcal{Z}(w_{n-1}, w_{n-2}) - \frac{\partial}{\partial \phi} R(w_n)^\top \phi \\ &= \underbrace{\left[\sum_j p(j | w_{n-1}, w_{n-2}) R(w_j) \right]}_{\text{model expected next word vector}} - \underbrace{R(w_n)}_{\text{data vector}} \end{aligned}$$

Learning the features: the log-bilinear language model

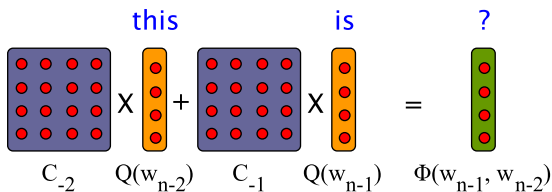


Error objective: $E = -\log p(w_n | w_{n-1}, w_{n-2})$

$$\frac{\partial}{\partial Q(j)} E = \frac{\partial \phi}{\partial Q(j)} \times \frac{\partial E}{\partial \phi}$$

$$\begin{aligned} \frac{\partial \phi}{\partial Q(j)} &= \frac{\partial}{\partial Q(j)} [C_{-2} Q(w_{n-2}) + C_{-1} Q(w_{n-1})] \\ &= \delta(j, w_{n-2}) C_{-2}^T + \delta(j, w_{n-1}) C_{-1}^T \end{aligned}$$

Learning the features: the log-bilinear language model



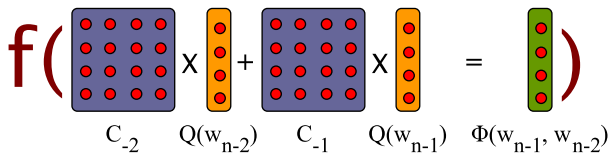
Error objective: $E = -\log p(w_n | w_{n-1}, w_{n-2})$

$$\frac{\partial}{\partial C_{-2}} E = \frac{\partial E}{\partial \phi} \times \frac{\partial \phi}{\partial C_{-2}}$$

$$\begin{aligned} \frac{\partial \phi}{\partial C_{-1}} &= \frac{\partial}{\partial A} [C_{-1} Q(w_{n-2}) + C_{-2} Q(w_{n-1})] \\ &= Q(w_{n-2})^T \end{aligned}$$

Adding non-linearities: the neural language model

Replacing the simple bi-linear relationship between context and output words with a more powerful non-linear function $f(\cdot)$ (logistic sigmoid, tanh, etc.):

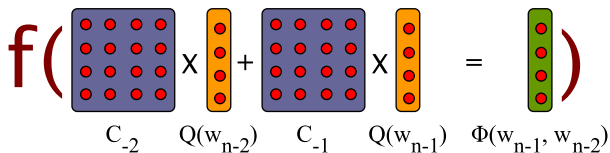

$$f\left(\begin{matrix} \text{grid } C_{-2} \\ \text{vector } Q(w_{n-2}) \end{matrix} \times + \begin{matrix} \text{grid } C_{-1} \\ \text{vector } Q(w_{n-1}) \end{matrix} \right) = \begin{matrix} \text{vector } \Phi(w_{n-1}, w_{n-2}) \end{matrix}$$

$$p(w_n | w_{n-1}, w_{n-2}) \propto \exp \left[R(w_n)^\top f \left(C^1 Q(w_{n-1}) + C^2 Q(w_{n-2}) \right) + b_{w_n} \right]$$

This is a neural language model!

Adding non-linearities: the neural language model

Replacing the simple bi-linear relationship between context and output words with a more powerful non-linear function $f(\cdot)$ (logistic sigmoid, tanh, etc.):


$$f\left(\begin{matrix} \text{4x4 grid} \\ C_{-2} \end{matrix} \times \begin{matrix} \text{4x1 column} \\ Q(w_{n-2}) \end{matrix} + \begin{matrix} \text{4x4 grid} \\ C_{-1} \end{matrix} \times \begin{matrix} \text{4x1 column} \\ Q(w_{n-1}) \end{matrix} = \begin{matrix} \text{4x1 column} \\ \Phi(w_{n-1}, w_{n-2}) \end{matrix} \right)$$

if $f =$ the element wise logistic sigmoid $\sigma(\cdot)$:

$$\begin{aligned} \frac{\partial}{\partial \phi} E &= \frac{\partial \sigma(\phi)}{\partial \phi} \circ \frac{\partial E}{\partial \sigma(\phi)} \\ &= \sigma(\phi)(1 - \sigma(\phi)) \circ \left[\sum_j p(j|w_{n-1}, w_{n-2}) R(w_j) - R(w_n) \right] \end{aligned}$$

where \circ is the element wise product.

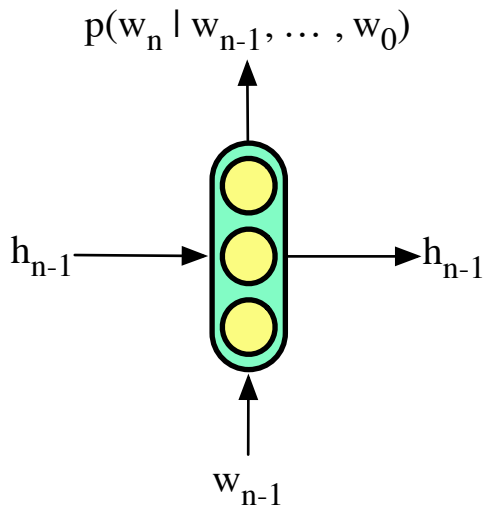
Infinite context: a recurrent neural language model

A recurrent LM drops the ngram assumption and directly approximate $p(w_n|w_{n-1}, \dots, w_0)$ using a recurrent hidden layer:

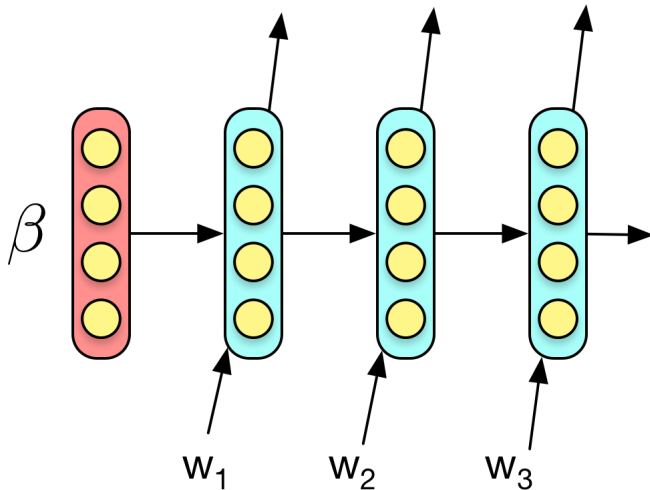
$$\begin{aligned}\phi^n &= C f(\phi^{n-1}) + WQ(w_{n-1}) \\ p(w_n|w_{n-1}, \dots, w_0) &\propto \exp \left[R(w_n)^\top f(\phi^n) + b_{w_n} \right]\end{aligned}$$

Simple RNNs like this are not actually terribly effective models. More compelling results are obtained with complex hidden units (e.g. Long Short Term Memory (LSTM) etc.), or by making the recurrent transformation C conditional on the last output.

Infinite context: a recurrent neural language model



Infinite context: a recurrent neural language model



LSTM LM

An LSTM cell,

$$x(t) = Q(w_{t-1}),$$

$$i(t) = \sigma(W_{xi}x(t) + W_{hi}h(t-1) + W_{ci}c(t-1) + b_i),$$

$$f(t) = \sigma(W_{xf}x(t) + W_{hf}h(t-1) + W_{cf}c(t-1) + b_f),$$

$$c(t) = f(t)c(t-1) + i(t) \tanh(W_{xc}x(t) + W_{hc}h(t-1) + b_c),$$

$$o(t) = \sigma(W_{xo}x(t) + W_{ho}h(t-1) + W_{co}c(t) + b_o),$$

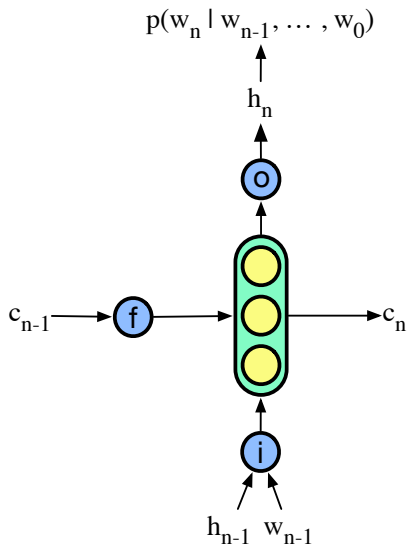
$$h(t) = o(t) \tanh(c(t)),$$

$$y(t) = W_y h(t) + b_y,$$

where $h(t)$ is the hidden state at time t , and i , f , o are the input, forget, and output gates respectively.

$$p(w_t | w_{t-1}, \dots, w_0) \propto \exp \left[R(w_t)^T y(t) + b_{w_t} \right]$$

Infinite context: a recurrent neural language model



Deep LSTM LM

A Deep LSTM cell with skip connections,

$$x'(t, k) = Q(w_{t-1}) || y'(t, k-1),$$

$$i(t, k) = \sigma (W_{kxi} x'(t, k) + W_{khi} h(t-1, k) + W_{kci} c(t-1, k) + b_{ki}),$$

$$f(t, k) = \sigma (W_{kxf} x'(t, k) + W_{khf} h(t-1, k) + W_{kcf} c(t-1, k) + b_{kf}),$$

$$c(t, k) = f(t, k) c(t-1, k) + i(t, k) \tanh (W_{kxc} x'(t, k) + W_{khc} h(t-1, k) + b_{kc}),$$

$$o(t, k) = \sigma (W_{kxo} x'(t, k) + W_{kho} h(t-1, k) + W_{kco} c(t, k) + b_{ko}),$$

$$h(t, k) = o(t, k) \tanh (c(t, k)),$$

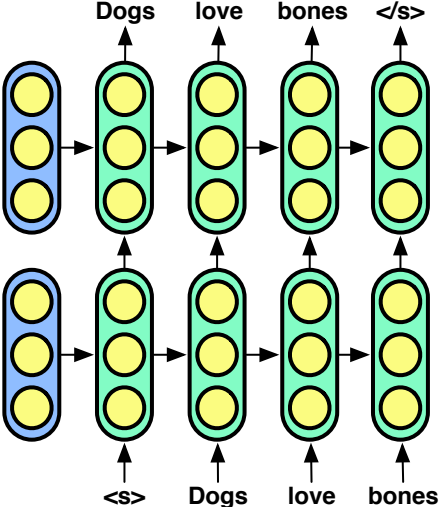
$$y'(t, k) = W_{ky} h(t, k) + b_{ky},$$

$$y(t) = y'(t, 1) || \dots || y'(t, K),$$

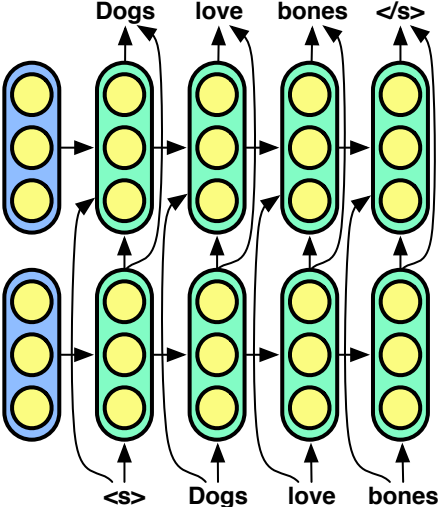
where $||$ indicates vector concatenation $h(t, k)$ is the hidden state for layer k at time t , and i, f, o are the input, forget, and output gates respectively.

$$p(w_t | w_{t-1}, \dots, w_0) \propto \exp [R(w_t)^T y(t) + b_{w_t}]$$

Deep LSTM LM



Deep LSTM LM



Efficiency

Most of the computational cost of a neural LM is a function of the size of the vocabulary and is dominated by calculating $R^T \phi$.

Efficiency

Most of the computational cost of a neural LM is a function of the size of the vocabulary and is dominated by calculating $R^T \phi$.

Solutions

Short-lists: use the neural LM for the most frequent words, and a traditional ngram LM for the rest. While easy to implement, this nullifies the neural LM's main advantage, i.e. generalisation to rare events.

Batch local short-lists: approximate the full partition function for data instances from a segment of the data with a subset of the vocabulary chosen for that segment.³

³On Using Very Large Target Vocabulary for Neural Machine Translation. Jean et al., ACL'15

Efficiency

Most of the computational cost of a neural LM is a function of the size of the vocabulary and is dominated by calculating $R^T \phi$.

Solutions

Approximate the gradient/change the objective: if we did not have to sum over the vocabulary to normalise during training it would be much faster. It is tempting to consider maximising likelihood by making the log partition function a separate parameter c , but this leads to an ill defined objective.

$$p_{\text{model}}(w_n | w_{n-1}, w_{n-2}, \theta) \equiv p_{\text{model}}^{\text{unnormalised}}(w_n | w_{n-1}, w_{n-2}, \theta) \times \exp(c)$$

Efficiency

Most of the computational cost of a neural LM is a function of the size of the vocabulary and is dominated by calculating $R^T \phi$.

Solutions

Approximate the gradient/change the objective: Mnih and Teh use noise contrastive estimation. This amounts to learning a binary classifier to distinguish data samples from (k) samples from a noise distribution (a unigram is a good choice):

$$p(\text{Data} = 1 | w_n, w_{n-1}, \theta) = \frac{p_{\text{model}}(w_n | w_{n-1}, \theta)}{p_{\text{model}}(w_n | w_{n-1}, \theta) + k p_{\text{noise}}(w_n)}$$

Now parametrising the log partition function as c does not degenerate. This is very effective for speeding up training, but has no impact on testing time.³

³In practice fixing $c = 0$ is effective. It is tempting to believe that this noise contrastive objective justifies using unnormalised scores at test time. This is not the case and leads to high variance results.

Efficiency

Most of the computational cost of a neural LM is a function of the size of the vocabulary and is dominated by calculating $R^T \phi$.

Solutions

Factorise the output vocabulary: One level factorisation works well (Brown clustering is a good choice, frequency binning is not):

$$p(w_n|\phi) = p(\text{class}(w_n)|\phi) \times p(w_n|\text{class}(w_n), \phi),$$

where the function $\text{class}(\cdot)$ maps each word to one class. Assuming balanced classes, this gives a $\sqrt{|\text{vocab}|}$ speedup.

This renders properly normalised neural LMs fast enough to be directly integrated into an MT decoder.³

³Pragmatic Neural Language Modelling in Machine Translation. Baltescu and Blunsom, NAACL'15

Efficiency

Most of the computational cost of a neural LM is a function of the size of the vocabulary and is dominated by calculating $R^T \phi$.

Solutions

Factorise the output vocabulary: By extending the factorisation to a binary tree (or code) we can get a $\log |\text{vocab}|$ speedup,³ but choosing a tree is hard (frequency based Huffman coding is a poor choice):

$$p(w_n | \phi) = \prod_i p(d_i | r_i, \phi),$$

where d_i is i^{th} digit in the code for word w_n , and r_i is the feature vector for the i^{th} node in the path corresponding to that code.

³A scalable hierarchical distributed language model. Mnih and Hinton, NIPS'09.

Comparison with traditional n-gram LMs

Good

- Better generalisation on unseen ngrams, poorer on seen ngrams. Solution: direct (linear) ngram features mimicking original log-linear language model features.
- Simple NLMs are often an order magnitude smaller in memory footprint than their vanilla ngram cousins (though not if you use the linear features suggested above!).

Bad

- Ngram NLMs are not as effective for extrinsic tasks such as Machine Translation compared to Kneser-Ney models, even when their intrinsic perplexity is much lower.
- NLMs easily beat Kneser-Ney models on perplexity for small training sets (<100M), but the representation size must grow with the data to be competitive at a larger scale.

Learning better representations for rich morphology

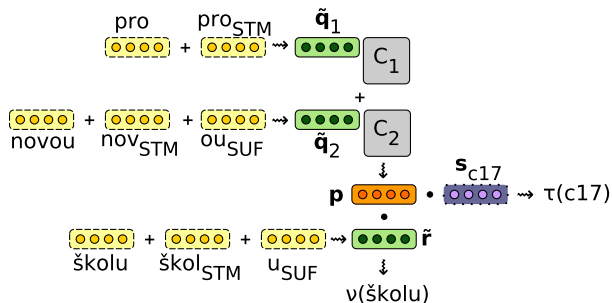
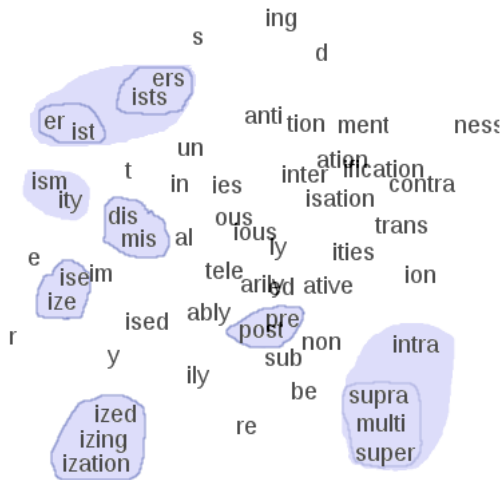


Illustration of how a 3-gram morphologically factored neural LM model treats the Czech phrase “pro novou školu” (for [the] new school).⁴

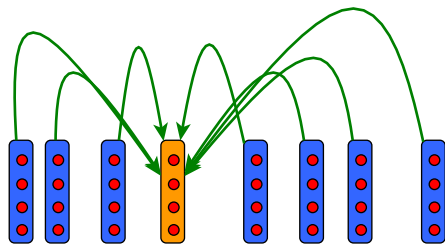
⁴Compositional Morphology for Word Representations and Language Modelling. Botha and Blunsom, ICML'14

Learning better representations for rich morphology



Learning representations directly

Collobert and Weston, Mikolov et al. word2vec, etc.

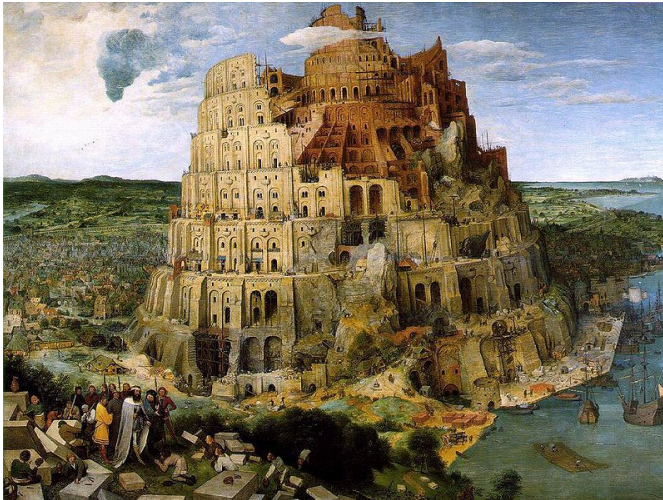


I got the shotgun. You got the briefcase.

If we do not care about language modelling, i.e. $p(\mathbf{w})$, and just want the word representations, we can condition on future context and/or use more efficient margin based objectives.

Intro to MT

The confusion of tongues:



Intro to MT: Language Divergence⁵

English	The plane is faster than the train.
Arabic	الطائرة أسرع من القطار (the-plane) (faster) (than) (the train)
Chinese	飞机比火车快 (plane) (compared-to) (train) (fast)
English	Who wrote this letter?
Arabic	من الذي كتب هذه الرسالة؟ (function-word) (who) (wrote) (this) (the-letter)
Chinese	这封信是谁写的？ (this) (letter) (be) (who) (write) (come-from) (function-word)

Models of translation

We observe an input sentence and require a model to produce its translation in another language:

Ich wollte dieses Buch lesen .



I wanted to read this book.

The set of possible output translations is, at best, exponential. We must structure our model according to a decomposition of the translation process.

MT History

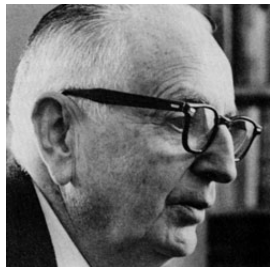
Warren Weaver memorandum, July 1949:



Thus it may be true that the way to translate from Chinese to Arabic, or from Russian to Portuguese, is not to attempt the direct route, shouting from tower to tower. Perhaps the way is to descend, from each language, down to the common base of human communication the real but as yet undiscovered universal language and then re-emerge by whatever particular route is convenient.

MT History

Warren Weaver memorandum, July 1949:



It is very tempting to say that a book written in Chinese is simply a book written in English which was coded into the Chinese code. If we have useful methods for solving almost any cryptographic problem, may it not be that with proper interpretation we already have useful methods for translation?

Parallel Corpora



Parallel Corpora

欢迎来到联合国

联合国
或联合国人民，团结起来，追求更美好的世界！

和平与安全 发展 人权 人道主义事务 国际法

你的联合国

联合国一览
联合国宪章
结构与结构
会员国
信息中心
常见问题
主要机关

新闻头条

会议、活动与展览

资源与服务

联合国入门

Welcome to the United Nations

UNITED NATIONS
We the peoples... A stronger UN for a better world.

Peace and Security Development Human Rights Humanitarian Affairs International Law

Year United Nations

In Focus

Main Bodies

In the News

Conferences, Meetings and Events

New on this site

100s of million words of translated text available for some language pairs, and billions of words of original text, more than an educated person would read in a lifetime.

MT History: Statistical MT at IBM

Fred Jelinek, 1988:



“Every time I fire a linguist, the performance of the recognizer goes up.”

MT History: Statistical MT at IBM

COMMENTS FOR THE AUTHOR(S) (clearness of presentation, lack of needed material or references to relevant work of other authors, language, etc; when rejection, the reasons should be given in detail):

The validity of statistical (information theoretic) approach to MT has indeed been recognized, as the authors mention, by Weaver as early as 1949.

And was universally recognized as mistaken by 1950.

(cf. Hutchins, MT: Past, Present, Future, Ellis Horwood, 1986, pp. 30ff. and references therein,

The crude force of computers is not science. The paper is simply beyond the scope of COLING.

Models of translation

The Noisy Channel Model

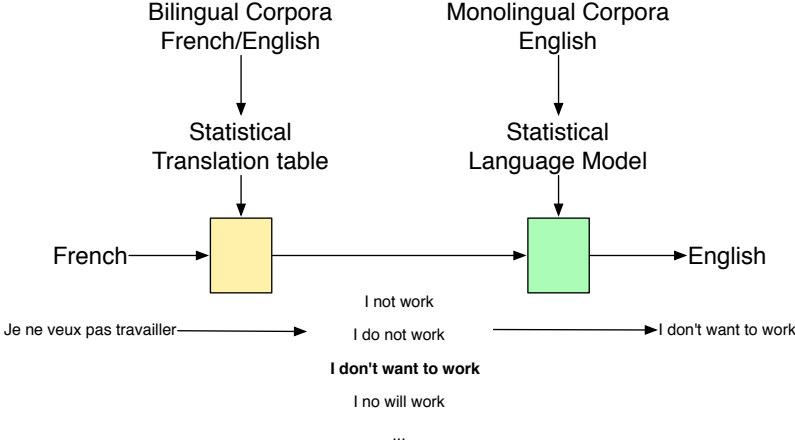
$$P(\text{English}|\text{French}) = \frac{P(\text{English}) \times P(\text{French}|\text{English})}{P(\text{French})}$$

$$\operatorname{argmax}_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}) = \operatorname{argmax}_{\mathbf{e}} [P(\mathbf{e}) \times P(\mathbf{f}|\mathbf{e})]$$

- Bayes' rule is used to reverse the translation probabilities
- the analogy is that the French is English transmitted over a *noisy channel*
- we can then use techniques from statistical signal processing and decryption to translate

Models of translation

The Noisy Channel Model



IBM Model 1: The first translation attention model!

A simple generative model for $p(\mathbf{s}|\mathbf{t})$ is derived by introducing a latent variable \mathbf{a} into the conditional probability:

$$p(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \frac{p(J|I)}{(I+1)^J} \prod_{j=1}^J p(s_j|t_{a_j}),$$

where:

- \mathbf{s} and \mathbf{t} are the input (source) and output (target) sentences of length J and I respectively,
- \mathbf{a} is a vector of length J consisting of integer indexes into the target sentence, known as the alignment,
- $p(J|I)$ is not important for training the model and we'll treat it as a constant ϵ .

To learn this model we use the EM algorithm to find the MLE values for the parameters $p(s_j|t_{a_j})$.

Models of translation

Phrase-based translation

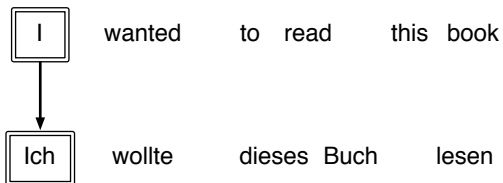
I wanted to read this book

Ich wollte dieses Buch lesen

Current state of the art: map larger sequences of words

Models of translation

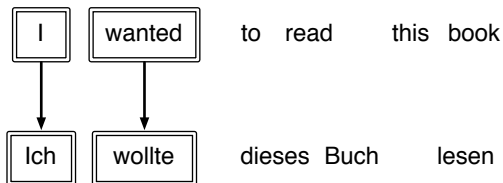
Phrase-based translation



Current state of the art: map larger sequences of words

Models of translation

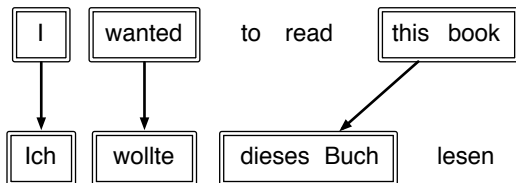
Phrase-based translation



Current state of the art: map larger sequences of words

Models of translation

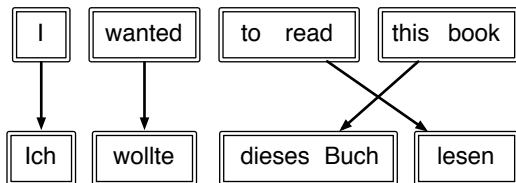
Phrase-based translation



Current state of the art: map larger sequences of words

Models of translation

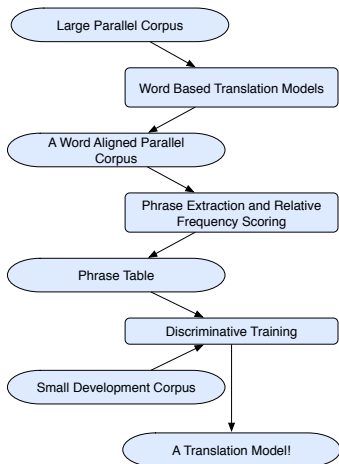
Phrase-based translation



Current state of the art: map larger sequences of words

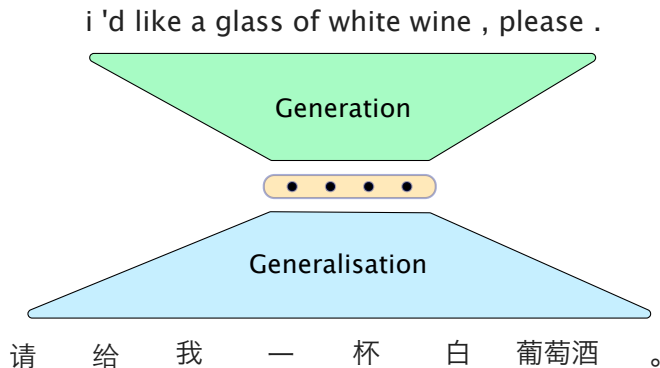
Models of translation

Phrase-based translation

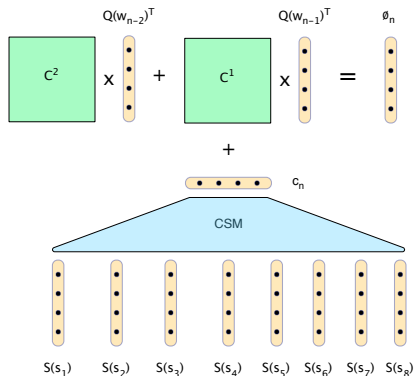


Current state of the art: map larger sequences of words

Encoder-Decoders⁶

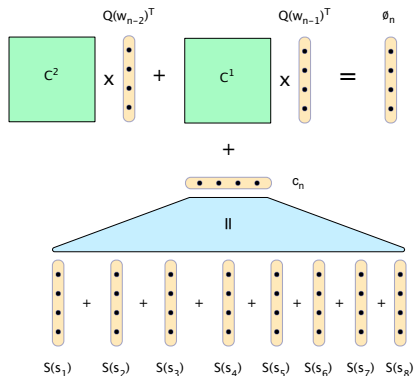


Encoder-Decoders



$$\phi_n = C^{-2}Q(w_{n-2}) + C^{-1}Q(w_{n-1}) + CSM(n, \mathbf{s})$$
$$p(w_n | w_{n-1}, w_{n-2}, \mathbf{s}) \propto \exp(R(w_n)^T \sigma(\phi_n) + b_{w_n})$$

Encoder-Decoders: A naive additive model



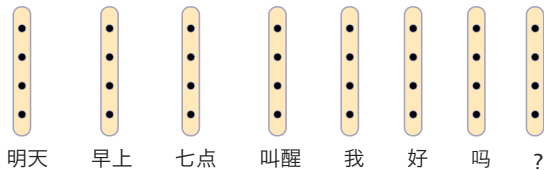
$$p_n = C^{-2}Q(w_{n-2}) + C^{-1}Q(w_{n-1}) + \sum_{j=1}^{|\mathbf{s}|} S(s_j)$$

$$p(w_n | w_{n-1}, w_{n-2}, \mathbf{s}) \propto \exp \left(R(w_n)^T \sigma(\phi_n) + b_{w_n} \right)$$

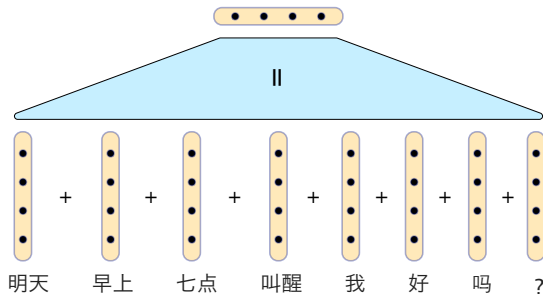
Encoder-Decoders: A naive additive model

明天 早上 七点 叫醒 我 好 吗 ？

Encoder-Decoders: A naive additive model

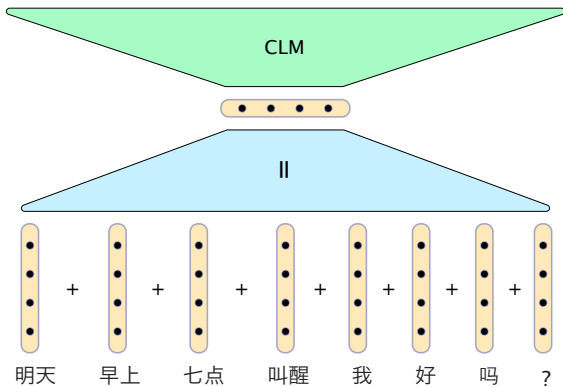


Encoder-Decoders: A naive additive model



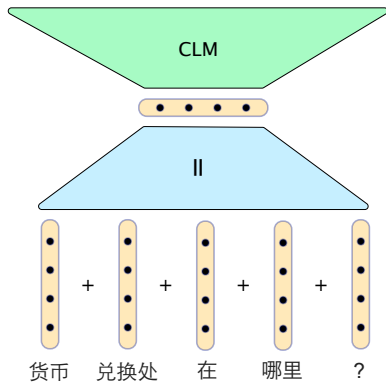
Encoder-Decoders: A naive additive model

may i have a wake-up call at seven tomorrow morning ?

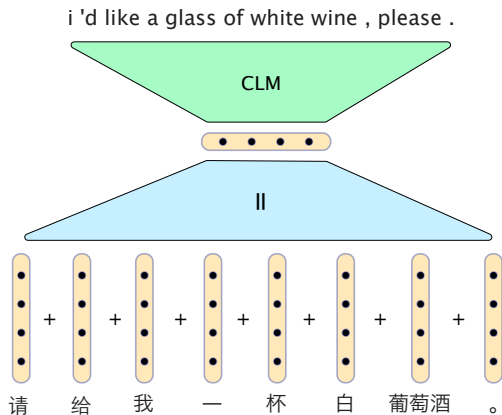


Encoder-Decoders: A naive additive model

where 's the currency exchange office ?

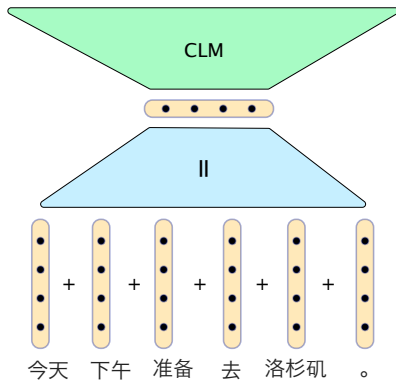


Encoder-Decoders: A naive additive model

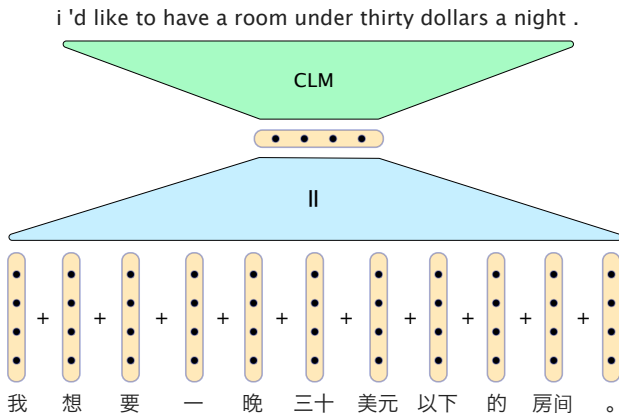


Encoder-Decoders: A naive additive model

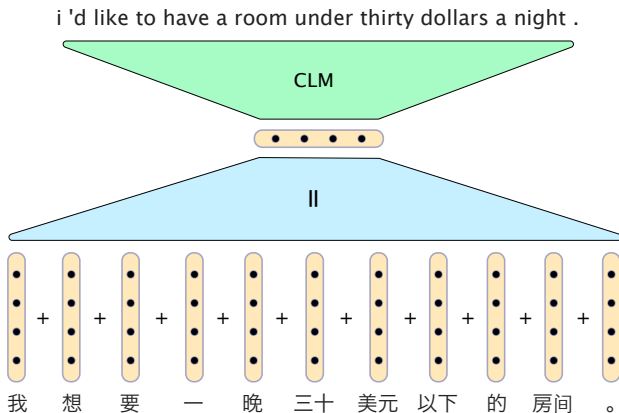
i 'm going to los angeles this afternoon .



Encoder-Decoders: A naive additive model



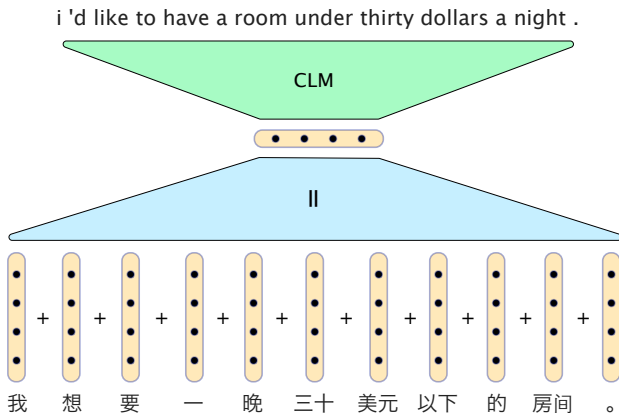
Encoder-Decoders: A naive additive model



Rough Gloss

I would like a night thirty dollars under room.

Encoder-Decoders: A naive additive model

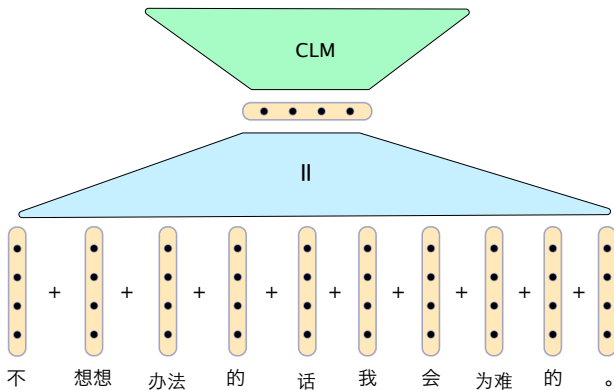


Google Translate

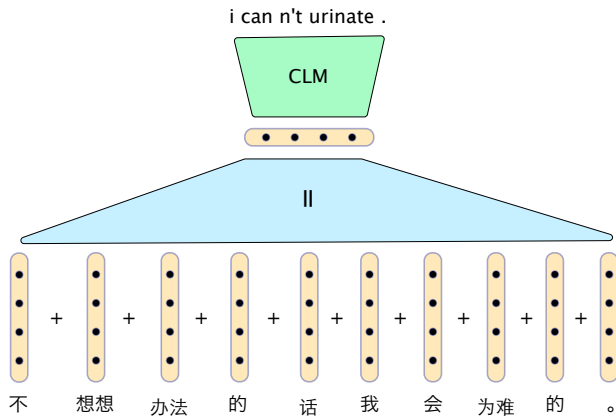
I want a late thirties under \$'s room.

Encoder-Decoders: A naive additive model

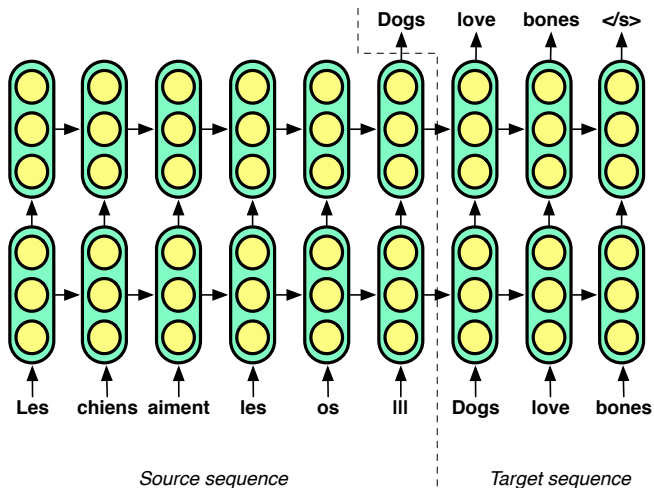
you have to do something about it .



Encoder-Decoders: A naive additive model

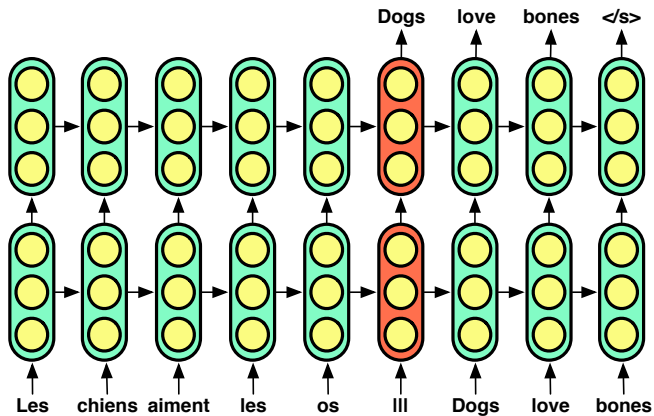


Recurrent Encoder-Decoders for MT⁷



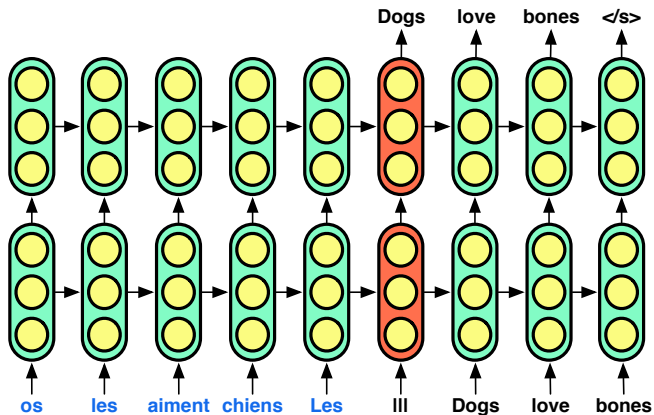
⁷Sequence to Sequence Learning with Neural Networks. Sutskever et al., NIPS'14

Recurrent Encoder-Decoders for MT⁷



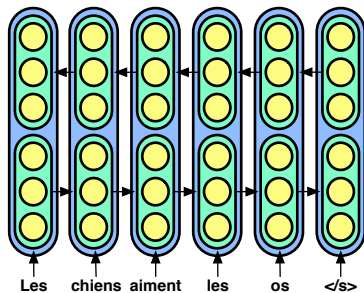
⁷Sequence to Sequence Learning with Neural Networks. Sutskever et al., NIPS'14

Recurrent Encoder-Decoders for MT⁷



⁷Sequence to Sequence Learning with Neural Networks. Sutskever et al., NIPS'14

Attention Models for MT⁸

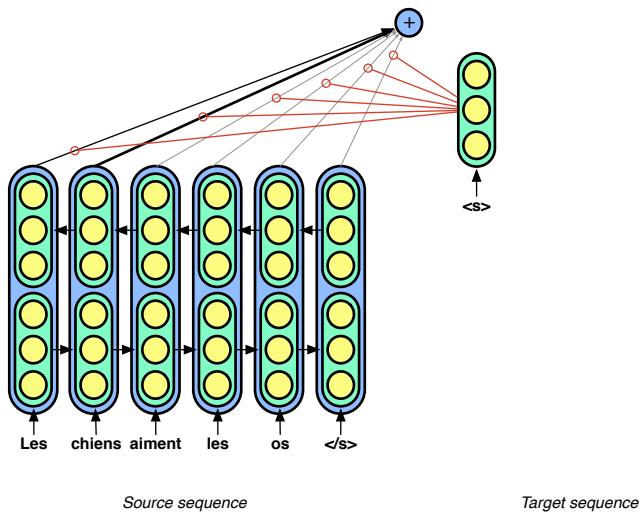


Source sequence

Target sequence

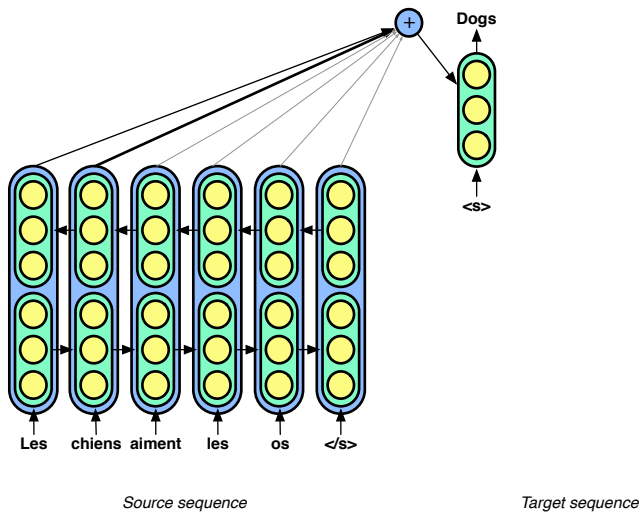
⁸Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau et al., ICLR'15

Attention Models for MT⁸



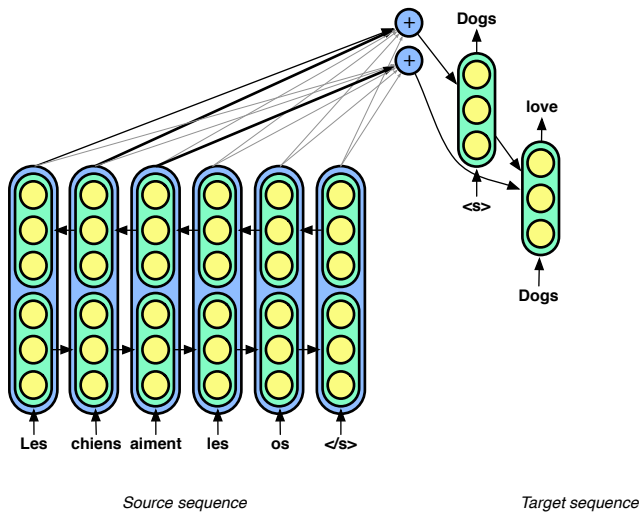
⁸Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau et al., ICLR'15

Attention Models for MT⁸



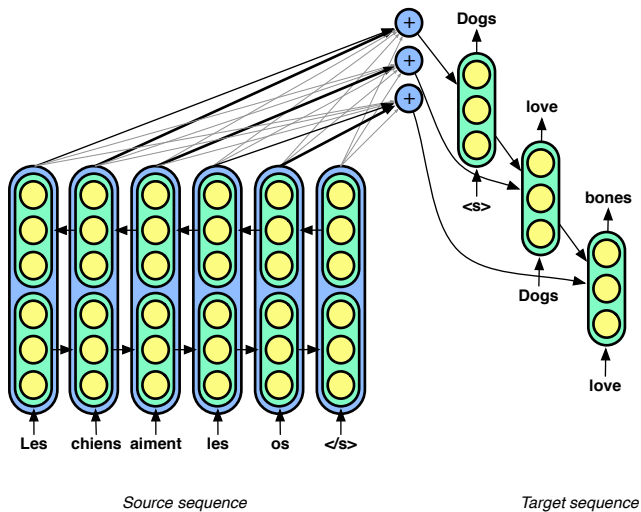
⁸Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau et al., ICLR'15

Attention Models for MT⁸



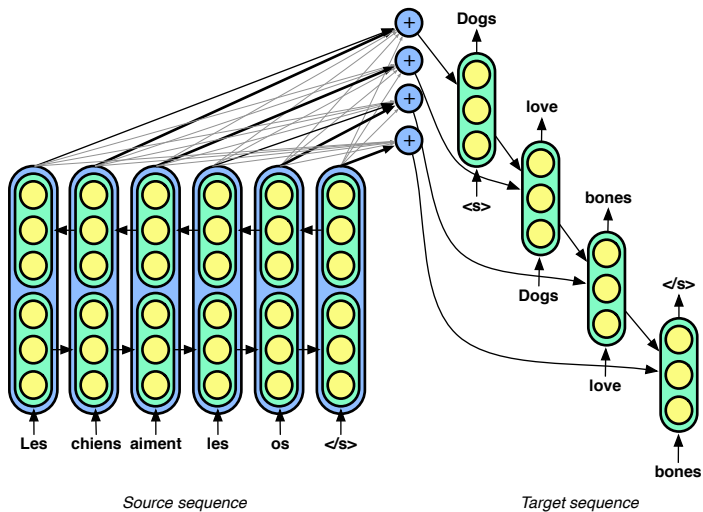
⁸Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau et al., ICLR'15

Attention Models for MT⁸



⁸Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau et al., ICLR'15

Attention Models for MT⁸



⁸Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau et al., ICLR'15

Montreal WMT Bleu Scores

(b) English→German (WMT-15)

Model	Note
24.8	Neural MT
24.0	U.Edinburgh, Syntactic SMT
23.6	LIMSI/KIT
22.8	U.Edinburgh, Phrase SMT
22.7	KIT, Phrase SMT

(c) English→Czech (WMT-15)

Model	Note
18.3	Neural MT
18.2	JHU, SMT+LM+OSM+Sparse
17.6	CU, Phrase SMT
17.4	U.Edinburgh, Phrase SMT
16.1	U.Edinburgh, Syntactic SMT

(d) German→English (WMT-15)

Model	Note
29.3	U.Edinburgh, Phrase SMT
29.1	KIT, Phrase SMT
28.9	Aachen, Phrase SMT
28.7	JHU, Phrase SMT
28.7	U.Edinburgh, Syntax SMT
27.9	Neural MT

(e) Czech→English (WMT-15)

Model	Note
26.2	JHU, Phrase SMT
24.5	U.Edinburgh, Syntax SMT
23.8	Neural MT
20.4	Dublin, Phrase SMT
14.5	Illinois

Attention based neural MT is competitive with other state of the art academic systems translating out of English. Augmenting with a language model improves performance in to English.

Issues, advantages and the future of MT

Summary

- The easiest gains from neural translation models can be had by incorporating them into a phrase based architecture,
- The full potential of neural translation models lie in their ability to capture multiple levels of representation, morphological, syntactic, and discourse dependencies.

Future

Since the 1980s research in Machine Translation has followed developments in Speech Recognition with a lag of approximately 5 years. There is not reason to doubt that this pattern will continue.

Google DeepMind and Oxford University



Google DeepMind



DEPARTMENT OF
**COMPUTER
SCIENCE**