# COLX-531: Neural Machine Translation

**Muhammad Abdul-Mageed**

muhammad.mageed@ubc.ca

**Deep Learning & NLP Lab**

The University of British Columbia

## Table of Contents

## Motivations

### What is more probable?

- Brewing a great cup of coffee/tea/espresso . . .
- Brewing a great cup of **entropy** . . .

### Language Models

- **Assign probabilities to sequences of words** (or characters, etc.).
- Play a **very significant role in NLP**.
- Classically, they are motivated by usage in tasks like **handwriting recognition, spelling correction, speech recognition**, and **machine translation**.
- Recently, their applications are exploding, including because of being an important block in learning (contextualized) word vectors **(referred to as unsupervised pre-training or generative pre-training)** . . .

## Statistical Language Models

- **A statistical LM**: The conditional probability of the next word given all the previous words.
- **Note:** We can also train bidirectionally (as in ELMo), or mask words (as in BERT). More later...

**1: Statistical LM**

$$\hat{P}(W_1^T) = \prod_{t=1}^{T} \hat{P}(w_t | w_1^{t-1})$$

**Brewing a great cup of coffee**

- coffee ($w_t$)
- Brewing a great cup of ($w_1^{t-1}$)
- Brewing a great cup of coffee ($W_1^T$)

## n-gram Language Models

- **Markov assumption**: We do not have to look too far in the past

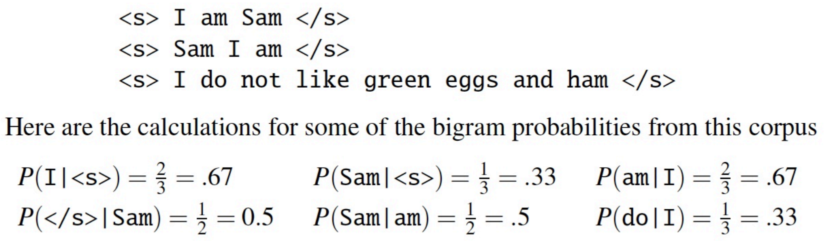
**2: n-gram LM**

$$\hat{P}(w_t|w_1^{t-1}) \approx \hat{P}(w_t|w_{t-N+1}^{t-1})$$

**3: Bigram LM (one word in the past)**

$$\hat{P}(w_t|w_{t-1}) = \hat{P}(w_t|w_{t-1})$$

## A Simple LM



```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

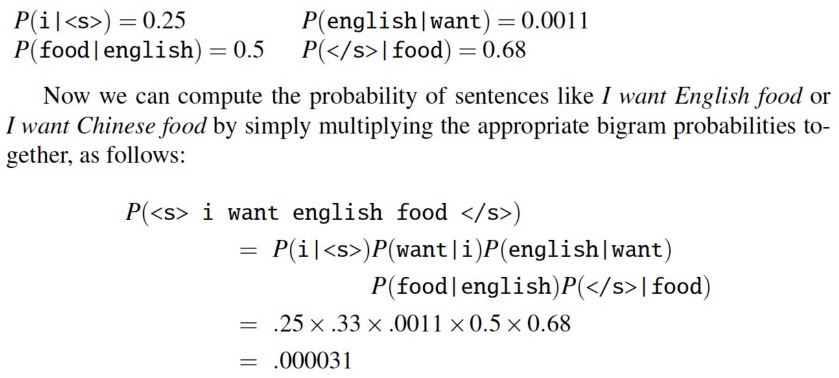
Here are the calculations for some of the bigram probabilities from this corpus

$P(\texttt{I}|\texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam}|\texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am}|\texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>}|\texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam}|\texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do}|\texttt{I}) = \frac{1}{3} = .33$

Figure: A simple LM on a mini-corpus. [From J&M, 2017, ch. 3]

# Computing P(sequence)



$P(\texttt{i}|\texttt{<s>}) = 0.25$ $\quad$ $P(\texttt{english}|\texttt{want}) = 0.0011$
$P(\texttt{food}|\texttt{english}) = 0.5$ $\quad$ $P(\texttt{</s>}|\texttt{food}) = 0.68$

Now we can compute the probability of sentences like *I want English food* or *I want Chinese food* by simply multiplying the appropriate bigram probabilities together, as follows:

$$\begin{aligned}
P(\texttt{<s> i want english food </s>}) &= P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\qquad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&= .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&= .000031
\end{aligned}$$

Figure: Computing P(sequence) with an LM. [See details in J&M, 2017, ch. 3]

## Notes on Computing Probs from LM

- We represent LM probabilities as **log probabilities**
- This **avoids numerical underflow** (if we were to use raw format)
- **Adding in log space = multiplying in linear space**
- To convert back, we exponentiate:

**4: Log Probabilities**

$$p_1 * p_2 * p_3 = \mathbf{exp}(log_2(p_1) + log_2(p_2) + log_2(p_3))$$

## Evaluating LMs

### Intrinsic vs. Extrinsic Evaluation

- Many models in NLP can be evaluated **intrinsically** and **extrinsically**
- **Intrinsic evaluation:** We use **perplexity**
- **Extrinsic evaluation**: Plugging LMs in an application like a **speech recognizer** or **MT system** is best method
- Again, modern language models are everywhere. Think about **ELMo** and **BERT**, for example.
- We need to split our data, possibly into 80% train, 10% dev, and 10% test.
- So, **don't train your LM on data that is part of your downstream task**. (Some popular papers unfortunately trained an LM on Wikipedia and tested on tasks whose data come from Wikipedia.)

## Perplexity as a Branching Factor

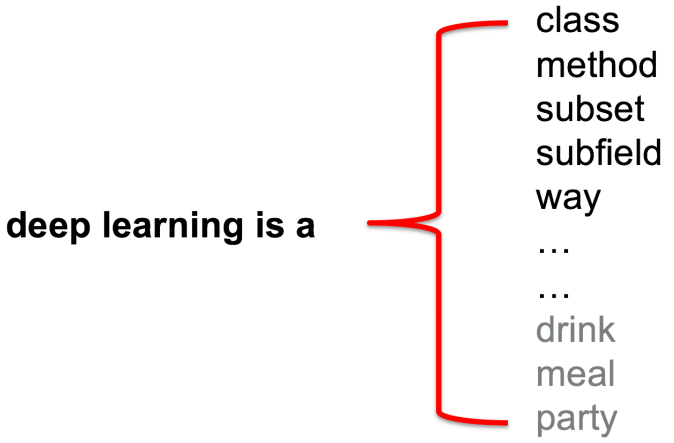


Figure: Tells us how many words can follow a given word. Lower is better. Why?

## Evaluation Intuition

### Perplexity

- **Perplexity** is the **weighted average branching factor**
- It is the **inverse probability on unseen data**, normalized by the number of words (for word-level perplexity).
- **Our goal is to maximize probability of unseen data**.
- Minimizing perplexity = maximizing probability.
- Perplexity is closely related to **entropy (recall: entropy is the avg amount of info.)**

### What's a good LM?

**One that best predicts unseen data** (test set). It's one that gives highest $P(S)$ for each sentence S in test.

## Perplexity

**5: Perplexity**

$$PP(W) = \hat{P}(w_1, w_2 \ldots w_T)^{-\frac{1}{T}}$$
$$= \sqrt[T]{\frac{1}{\hat{P}(w_1, w_2 \ldots w_T)}}$$

**6: Chain Rule**

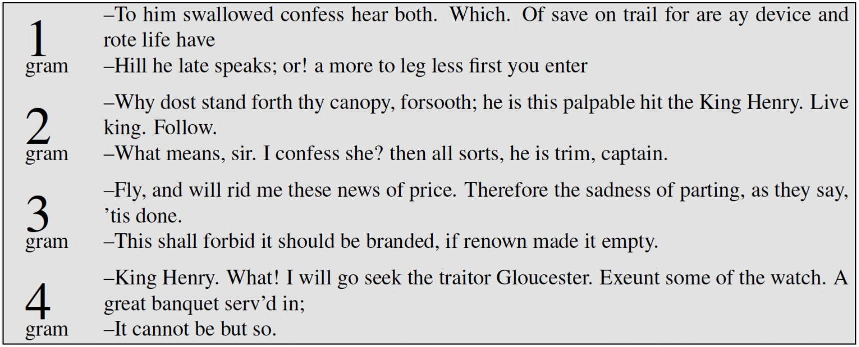
$$PP(W) = \sqrt[T]{\prod_{i=1}^{T} \frac{1}{\hat{P}(w_i | w_1 \ldots w_{i-1})}}$$

## Perplexity For Bigrams

**7: Perplexity for Bigrams**

$$PP(W) = \sqrt[T]{\prod_{i=1}^{T} \frac{1}{\hat{P}(w_i | w_{i-1})}}$$

## Sample Generation From Shakespeare

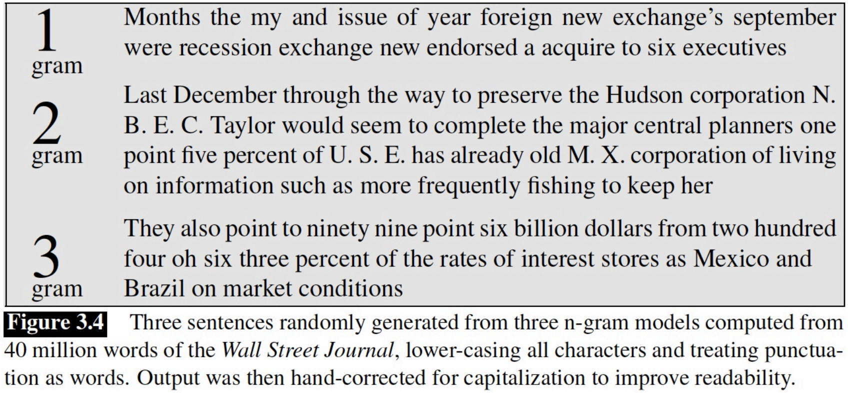


| | |
|---|---|
| 1 gram | –To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have<br>–Hill he late speaks; or! a more to leg less first you enter |
| 2 gram | –Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.<br>–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.<br>–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;<br>–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.

Figure: [From J&M, 2017, ch. 3]

## Sample Generation From WSJ



| | |
|---|---|
| 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.

Figure: [From J&M, 2017, ch. 3]

## Unseen words

### Smoothing

- We will see words at test time that we have not seen in **train**
- Can't assign these **zero probability** (otherwise we'll have division by zero!)
- Called **out-of-vocabulary (OOV)**
- We have to do some **smoothing**, e.g.:
  - Laplace smoothing (add-one)
  - Add-K smoothing
  - back-off and interpolation smoothing
  - Kneser-Ney smoothing
  - ...
- For details, see J&M (2017, ch. 03) ...

## Problems with n-gram LM

- **Limited to a short sub-sequence** (e.g., 2 words, for n=3).

**Data scarcity**

LMs for larger n-gram suffer from **data scarcity**.

- **Does not easily capture word "similarity".**

**Challenge with word similarity**

Consider the following two sentences where "cat" and "dog" have similar semantic and grammatical roles. [Bengio et al., 2003].

1. "The cat is walking in the bedroom"
2. "A dog was running in a room"

## Language Models in Continuous Space

- Bengio et al. (2003, p. 1139. JML paper):

### LM Via A Neural Net

- Express each **word as a feature vector** (real-valued).
- Express the **joint probability function** of word sequences in terms of these word vectors.
- **Learn the word vectors and the joint probability function simultaneously** (using a neural network).

## Why Does it Work?

- It is possible to naturally generalize (i.e., transfer probability mass) from (1) to (2-5) such that dog and cat end up with similar vectors:

**Example**

1. `The cat is walking in the bedroom`
2. `A dog was running in a room`
3. `The cat is running in a room`
4. `A dog is walking in a bedroom`
5. `The dog was walking in the room`
6. ...

# Bengio et al., 2003 Neural Model

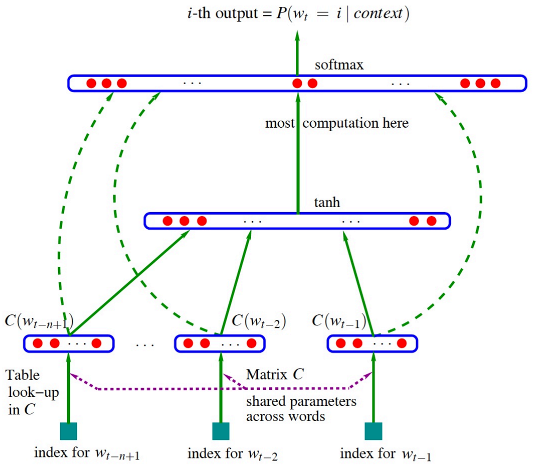


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where $g$ is the neural network and $C(i)$ is the $i$-th word feature vector.

## It Still Doesn't Scale. . .

It is such a great idea, but there is a **bottleneck**. . .

- **Computationally costly** to obtaining the output probabilities (since **softmax operates on all vocabulary**).
- **Bottleneck** in the computation of **the activations of the output layer**.
- An n-gram model **does not require the computation of the probabilities for all the words in the vocabulary**.
- **Mikolov et al. (2013)** Scale learning word vectors with a large vocabulary (V).

## LM & MT Model

**8: LM & MT Model**

$$\textbf{LM} = P(w_1, w_2, \ldots, w_T)$$

$$\textbf{MT Model} = P(y_1, y_2, \ldots, y_T | x)$$

**Compare LM & MT Model**

- **MT Model**: Generative model.
- Language model conditioned on source sentence.
- i.e., **Conditional language model**

## Different Ways to Generate Translations . . .

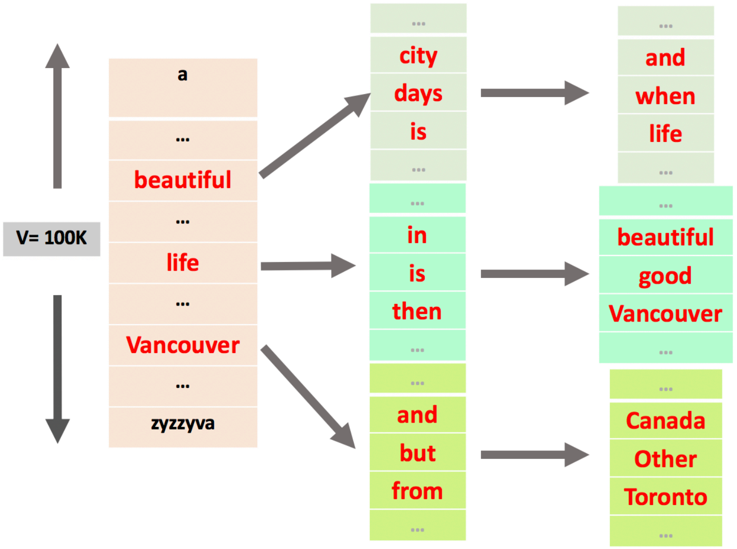
**La vie est belle à Vancouver.**

- Life →
  1. *is* . . .
     1. *beautiful* . . .
     2. *good* . . .
  2. *in* . . .
     1. *Vancouver* . . .

**La vie est belle à Vancouver.**

- Beautiful →
  1. *is* . . .
     1. *life* . . .
  2. *in* . . .
     1. *Vancouver* . . .

## Beam Search – Beam Width $B = 3$



## Beam Search, More Fromally

### 9: Beam Search

$$\underset{y}{\operatorname{argmax}} \prod_{t=1}^{T_y} P(y_t | x, y_1, \ldots y_{t-1})$$

– **Take log and sum up. Why? To avoid numerical underflow.**

$$\underset{y}{\operatorname{argmax}} \sum_{t_1}^{T_y} P(y_t | x, y_1, \ldots y_{t-1})$$

## Length Normalization

### Problem

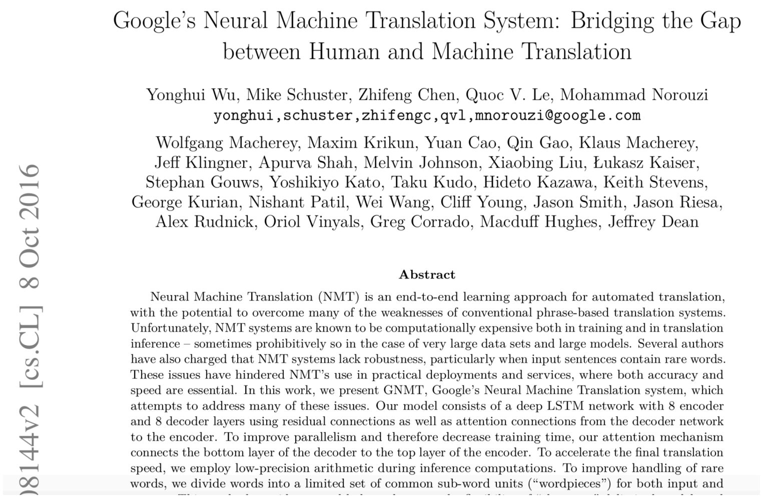
- **Longer sentences will have smaller probailities**.
- This is due to multiplying. (or taking log of a prob.:(always $\leq 1$).
- **Solution**: Normalize by sentence length

### 10: Length Normalization

$$\frac{1}{T_y^{\alpha}} \underset{y}{\operatorname{argmax}} \sum_{t_1}^{T_y} P(y_t | x, y_1, \ldots y_{t-1})$$

– $\alpha$ is how much we normalize. ($\alpha = 1$: fully normalizing by length. $\alpha = 0$: Not normalizing at all)

## Google NMT System



# Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

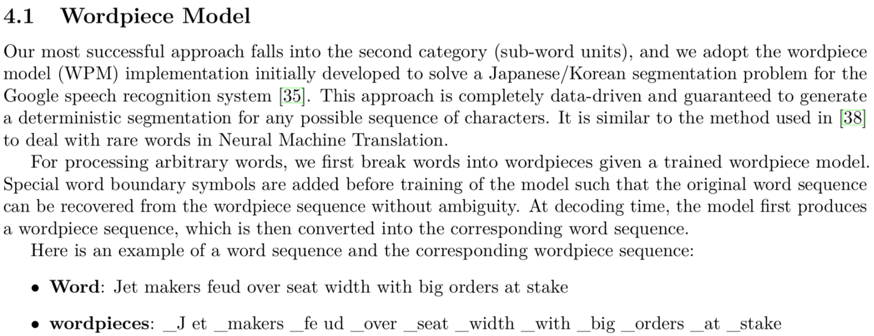
Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
`yonghui,schuster,zhifengc,qvl,mnorouzi@google.com`

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

**Abstract**

Neural Machine Translation (NMT) is an end-to-end learning approach for automated translation, with the potential to overcome many of the weaknesses of conventional phrase-based translation systems. Unfortunately, NMT systems are known to be computationally expensive both in training and in translation inference – sometimes prohibitively so in the case of very large data sets and large models. Several authors have also charged that NMT systems lack robustness, particularly when input sentences contain rare words. These issues have hindered NMT's use in practical deployments and services, where both accuracy and speed are essential. In this work, we present GNMT, Google's Neural Machine Translation system, which attempts to address many of these issues. Our model consists of a deep LSTM network with 8 encoder and 8 decoder layers using residual connections as well as attention connections from the decoder network to the encoder. To improve parallelism and therefore decrease training time, our attention mechanism connects the bottom layer of the decoder to the top layer of the encoder. To accelerate the final translation speed, we employ low-precision arithmetic during inference computations. To improve handling of rare words, we divide words into a limited set of common sub-word units ("wordpieces") for both input and

# Google NMT System *Contd.*



### 4.1 Wordpiece Model

Our most successful approach falls into the second category (sub-word units), and we adopt the wordpiece model (WPM) implementation initially developed to solve a Japanese/Korean segmentation problem for the Google speech recognition system [35]. This approach is completely data-driven and guaranteed to generate a deterministic segmentation for any possible sequence of characters. It is similar to the method used in [38] to deal with rare words in Neural Machine Translation.

For processing arbitrary words, we first break words into wordpieces given a trained wordpiece model. Special word boundary symbols are added before training of the model such that the original word sequence can be recovered from the wordpiece sequence without ambiguity. At decoding time, the model first produces a wordpiece sequence, which is then converted into the corresponding word sequence.

Here is an example of a word sequence and the corresponding wordpiece sequence:

- **Word**: Jet makers feud over seat width with big orders at stake
- **wordpieces**: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

## Google NMT System *Contd.*

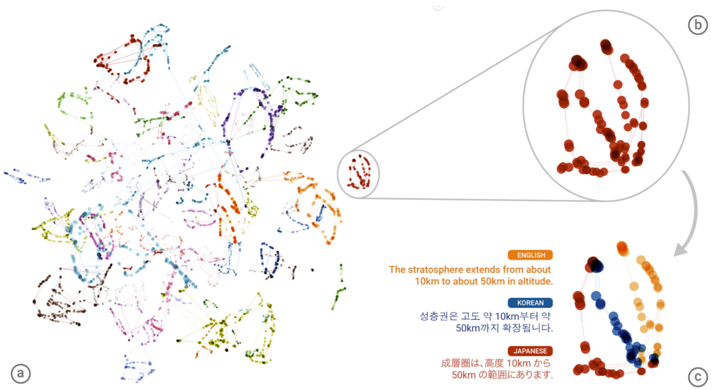
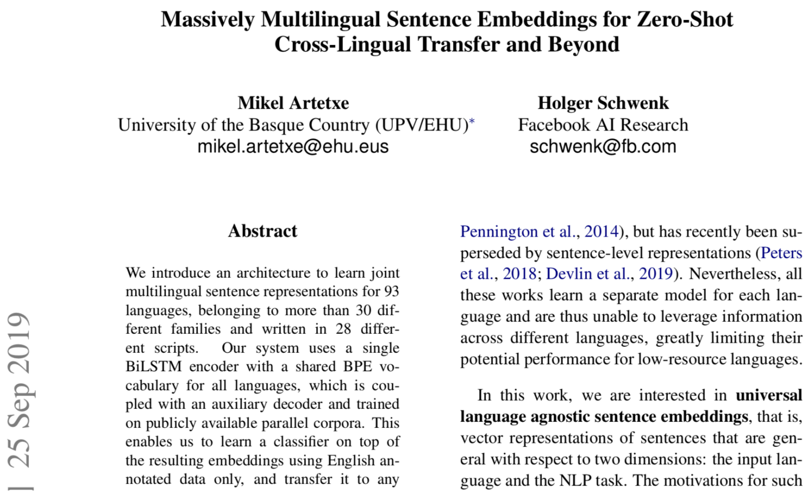


Figure 2: A t-SNE projection of the embedding of 74 semantically identical sentences translated across all 6 possible directions, yielding a total of 9,978 steps (dots in the image), from the model trained on English↔Japanese and English↔Korean examples. (a) A bird's-eye view of the embedding, coloring by the index of the semantic sentence. Well-defined clusters each having a single color are apparent. (b) A zoomed in view of one of the clusters with the same coloring. All of the sentences within this cluster are translations of "The stratosphere extends from about 10km to about 50km in altitude." (c) The same cluster colored by source language. All three source languages can be seen within this cluster.

## Massively Multilingual MT (Facebook)



**Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond**

**Mikel Artetxe**
University of the Basque Country (UPV/EHU)*
mikel.artetxe@ehu.eus

**Holger Schwenk**
Facebook AI Research
schwenk@fb.com

| 25 Sep 2019

**Abstract**

We introduce an architecture to learn joint multilingual sentence representations for 93 languages, belonging to more than 30 different families and written in 28 different scripts. Our system uses a single BiLSTM encoder with a shared BPE vocabulary for all languages, which is coupled with an auxiliary decoder and trained on publicly available parallel corpora. This enables us to learn a classifier on top of the resulting embeddings using English annotated data only, and transfer it to any

Pennington et al., 2014), but has recently been superseded by sentence-level representations (Peters et al., 2018; Devlin et al., 2019). Nevertheless, all these works learn a separate model for each language and are thus unable to leverage information across different languages, greatly limiting their potential performance for low-resource languages.

In this work, we are interested in **universal language agnostic sentence embeddings**, that is, vector representations of sentences that are general with respect to two dimensions: the input language and the NLP task. The motivations for such