Natural Language Understanding, Generation and Machine Translation - Week 6 - Open Vocabulary Models, Low Resource MT and Ethics

Antonio León Villares

April 2023

Based on:

- *Neural Machine Translation of Rare Words with Subword Units, by Sennrich et al.*
- *BPE-Dropout: Simple and Effective Subword Regularisation*
- *Survey of Low-Resource Machine Translation, by Haddow et al.*
- *THe Social Impact of Natural Language Processing, by Hovy and Spruit*

1 Open Vocabulary Models

1.1 The Problem with Closed Vocabulary Models

- **How do we currently represent text?**
  - we assume that all words at **training** time represent **all** possible words
  - in particular, we've taken the **input** and **output** of our models to come from some **fixed vocabulary**
  - currently, we learn a **vectorised** representation for words (either **one-hot** or **neural embeddings**), which is used by models to compute a **probability distribution** over all words in the vocabulary
  - these **closed vocabularies** are often very **large** (10,000 - 100,000) symbols, so:
    - training is **expensive** - large memory + large training time
    - **decoding** will also be expensive
- **Why is the closed vocabulary assumption not valid for NLU/NLG problems?**
  - **language-related** problems will be **open-vocabulary** problems by definition:
    1. Language is **dynamic**: new words are constantly generated, for example by joining old words together (i.e "antifragile "covidiot")
    2. **Proper names** (i.e countries, cities, people) or **numbers** are common, but are part of open word classes (i.e it is unlikely that N'Djaema is part of a vocabulary, even though it is a perfectly valid city name which is morphologically simple)
    3. Some languages are **agglutinative**: valid words can be generated by joining together many **morphemes**
  - this showcases that to build **robust** machine translation models, we need to shift towards **open vocabulary models**

1.2 Previous Approaches to Open Vocabulary Models

1.2.1 Ignoring Rare Words

- **What does this approach consist of?**
  - define a **closed vocabulary** with n words (for example to cover the top 95% most frequent words in training)
  - remaining words are replaced with a `UNK` token
- **Why is ignoring rare words insufficient?**
  - left-out words typically carry the **most important information**
  - for instance:

`SOURCE`:	Mr **Gallagher** has offered a ray of hope.
`REFERENCE`:	Herr **Gallagher** hat einen hoffnungsstrahl ausgesandt.
`TRANSLATION`:	Herr `UNK` hat einen hoffnungsstrahl ausgesandt.

1.2.2 Approximative Softmax

- **How are approximative softmax models trained?**
  - one of the biggest **bottlenecks** in NMT is **large vocabularies**, which make computations (particularly softmax) **expensive**
  - with **approximative softmax** (suggested in On Using Very Large Target Vocabulary for Neural Machine Translation), the training data gets **split**, and a **vocabulary** is generated for each split
  - this defines smaller closed vocabulary, which nonetheless covers the whole partition
  - **softmax** can then be computed over the splits, which is less computationally expensive than using the whole vocabulary
- **How can approximative softmax models be used at test time?**
  - it is likely that words at test time are spread out over many vocabulary partitions
  - hence, a new vocabulary is generated by using some crude method (i.e using a translation dictionary)
- **What are the 2 key limitations of approximative softmax?**
  1. **Not Open**: this allows computations over larger vocabularies, but if a word never appears in **training**, it will be completely unknown for the model
  2. **Rare Words**: these might be seen once or twice by the model, so it will have little knowledge of how to translate them

1.2.3 Back-Off Models

- **What problems do back-off models seek to solve?**
  - **back-off models** for translation were suggested in Addressing the Rare Word Problem in Neural Machine Translation
  - it seeks to prevent the use of `UNK` when translating rare words, such as proper names
- **How can backoff help reduce problems when translating rare words?**
  - during **training**, rare words are replaced by using `UNK`
  - when using the system, we can **align** unknown words with `UNK`, and use this to fill in the gaps for the `UNK`
- **What are the 4 limitations of the back-off model?**
  1. **1-1 Mapping**: the model assumes that 1 word in the origin corresponds to 1 word in the reference, so it doesn't take into account **compound words**

  2. **Transliteration**: for languages with different alphabets (i.e cyrillic, japanese), **transliteration** might be required to map between the two
  3. **Alignment**: the **alignment** might not be perfect

4. **Morphology**: morphological rich languages (like Turkish) might not work well for these models. Turkish words can have many different forms depending on the context, and there are a large number of inflectional and derivational morphemes that can be added to the root word to create new words.

1.3 Subword Neural Machine Translation

Subword NMT is currently the main approach to Open Vocabulary Models.

1.3.1 Motivation

- **Why can translation based on subwords be useful?**
  - the main problem with **neural machine translation** is that despite **huge vocabularies**, not every word can be accounted for
  - instead, we can try considering **morphemes** and **subwords**, since these can be combined to generate **unknwon words**:
    1. **Compounding**:

       `SOURCE` : they charge a carry-on bag fee
       `REFERENCE` : sie erheben eine Handgepäckgebühr

    2. **Names**: names are typically unseen, but by breaking words up into morphemes, we can even use **transliteration** to convert names between different languages

       `SOURCE` : Obama
       `REFERENCE` : Obama → Обама

    3. **Morphological Variation**: in agglutinative languages, we can generate new words by joining morphemes

       `ORIGINAL` : OSMANLILAŞTIRAMAYABİLECEKLERİIMİZDENMİSSİNİZ
       `SEGMENTED` : OSMAN-LI-LAŞ-TIR-AMA-YABİL-ECEK-LER-İIMİZ-DEN-MİSSİNİZ

    4. **Numbers**: things like numbers/dates/techincal terms might need to get written in a specific format:

       `SOURCE` : 10-12-2020
       `REFERENCE` : December 10 2020

1.3.2 Byte Pair Encoding for Segmentation

- **What features are desirable when segmenting words into subwords?**
  1. **Generalisable**: ultimately we want to tackle **Open Vocabulary Problems**, so the **segmentation** should be able to cover all words through a small enough vocabulary (even for **unseen** words)
  2. **Text Size**: if we use extremely high **granularity** (i.e segment at each character) input and output text will involve too many tokens, which can slow down learning/decoding; we need large enough morphemes which can nonetheless encompass every element of a vocabulary
- **What is the current approach to subword segmentation?**
  - approach proposed by Sennrich et al. in Neural Machine Translation of Rare Words with Subword Units

- to generate the **segmentations**, use **Byte Pair Encoding** (BPE) (proposed by Gage in A New Algorithm for Data Compression)
- also considered using character-level **n-grams** to break down words, but BPE worked best

- **How does BPE generate subword segmentations?**
  - we start by considering all **characters** as **subwords**
  - repeatedly replace the **most frequent** symbol pair "A B" with "AB":

$$L\ O \rightarrow LO$$
$$LO\ W \rightarrow LOW$$
$$E\ LOW \rightarrow ELOW$$

  - the only **hyperparamter** to this model is the **number of merges** to perform

word	freq
'l o w'	5
'l o w e r'	2
'n e w e s t'	6
'w i d e s t '	3

vocabulary:
l o w e r n s t i d

word	freq
'l o w'	5
'l o w e r'	2
'n e w **es** t'	6
'w i d **es** t '	3

vocabulary:
l o w e r n s t i d
es

word	freq
'l o w'	5
'l o w e r'	2
'n e w **est**'	6
'w i d **est** '	3

vocabulary:
l o w e r n s t i d
es **est**

word	freq
'**lo** w'	5
'**lo** w e r'	2
'n e w **est**'	6
'w i d **est** '	3

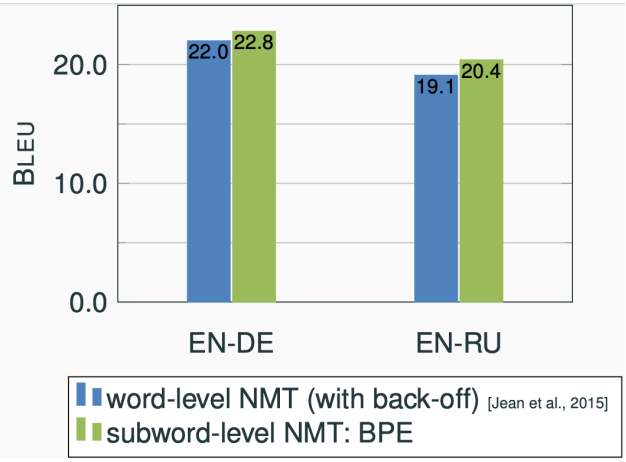
vocabulary:
l o w e r n s t i d
es est **lo**

. 1: With this vocabulary, even if we've never seen the word "lowest", we'd still be able to break it down into **known** subwords, namely lowest.

- with **BPE**, we can join characters which frequently occur together, which gives a good compromise between the granularity and length of the subwords, which makes the segmentations apply well, even to unknown words
- in practice, this means that the **most frequent** words will appear **whole** within the vocabulary, after which we'll start seeing **subwords**

1.3.3 Evaluating BPE

- **How does using subword NMT compare with using word-level NMT?**
  - in the experiments by Sennrich et al., they compared 2 models (subword vs word NMT) for English-German and English-Russian translation
  - subword NMT obtained a significatly higher `BLEU` in both tasks



. 2: Comparison between **BPE Subword NMT** and **Word-Level NMT with Backoff**. The model for translation was an attentional encoder-decoder network.

- **Do the segmentations generated by BPE correlate with "human-like" segmentations?**
  - not necessarily: BPE generates **segmentations** based solely on **statistics**
  - this means that **compounded words** won't necessarily be broken down into its components
  - nonetheless, this doesn't affect translation quality too much

`SOURCE :`	health research institutes
`REFERENCE :`	Gesundheitsforschungsinstitute
`WORD-LEVEL (BACKOFF) :`	Forschungsinstitute
`BPE :`	Gesundheits/forsch/ungsin/stitute

  - as we can see, BPE splits up "forschungs" and "institute", but nonetheless obtains a correct translation; the word-level translation chooses to ignore the "health" part, which is integral to meaning
- **How well do BPE segmentations deal with translating between different alphabets?**
  - if we just use **back-off**, this won't be able to **transliterate** between alphabets
  - even if we use **bigrams**, and transliterate these, the result won't always be correct

– however, BPE seems to handle these cases fine

SOURCE :	rakfisk
REFERENCE :	ракфиска
WORD-LEVEL (BACKOFF) :	rakfisk → UNK → rakfisk
BIGRAMS :	ra/kf/is/k → ра/кф/ис/к
BPE :	rak/f/isk → рak/ф/иска

- **How does BPE deal with segmentations in different languages?**
  - if we translate between 2 languages, we need to be able to **segment** both the **source** and the **reference**
  - in the examples above, **shared BPE** was used: develop BPE **segmentations** using both languages to initalise the vocabulary
  - in the case in which there's different alphabets (i.e English-Russian) we use a **romanised** russian (i.e ф → f, б → b, и → i, с → s)
- **Why is shared BPE better than using 2 different segmentations for the different languages?**
  - using **shared BPE** ensures that the model learns a more **robust** understanding of **language structure** across the different languages
  - the model will get a better understanding of which subwords are **frequent** between the different languages, thus allowing it to better **align** these subwords, obtaining more **consistent** results
  - moreover, this allows the model to learn the **same** embedding for **subwords** across the different languages, simplifying the model
  - **shared BPE** has also been used for **multilingual translation**, where we might have 10-15 different languages, and we don't want to have a huge separate vocabulary for each language
- **How does using shared and separate BPE affect the segmentations found by BPE?**
  - if we train **separate BPE** for English and Russian, we obtain an **inconsistent segmentation** between the languages:

    rak/f/isk → пра/ф/иск
  - here, "rak"seems to get translated into "pra → пра"
  - this happens because when using **separate BPE** in training, we obtain pairs which look like:

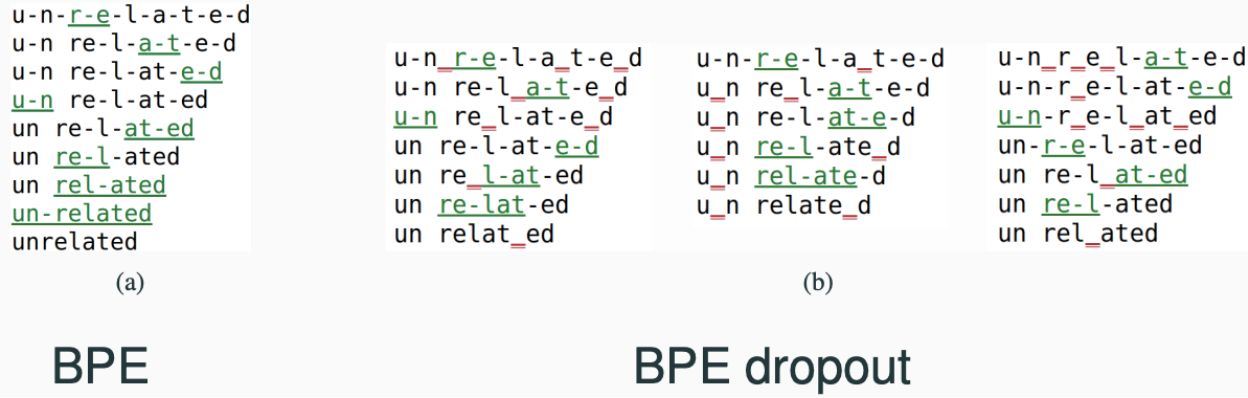
    p/rak/ri/ti → пра/крит/и

    so whilst in the source language "p" doesn't typically prefix "rak", in Russian, a lot of words must begin with "pra → пра", which means that source and reference have a **different** number of segmentations, which difficults a correct alignement and translation
  - if we instead use **shared BPE**, these inconsistencies disappear:

    pra/krit/i → пра/крит/и
- **Why is BPE still not ideal?**
  - BPE is essentially **hand-engineering** the segmentations
  - hence for these models, we still require a separate **preprocessing** step
  - ideally, the **translation model** should learn to **automatically** make these segmentations, whilst learning to translate sentences

1.3.4 BPE-Dropout

- **What is BPE-Dropout?**
  - an adaptation for BPE, aimed at **regularising** the **subword segmentation**
  - "forget"to combine the most **frequent** words with a given probability
  - for instance, if we have "un-rel-at-ed" , instead of merging "un" or "ed", it might choose to merge "rel-at" → "relat" instead



. 3: hyphens denote possible merges. in green, all the merges which were performed, whilst in red the merges which were dropped.

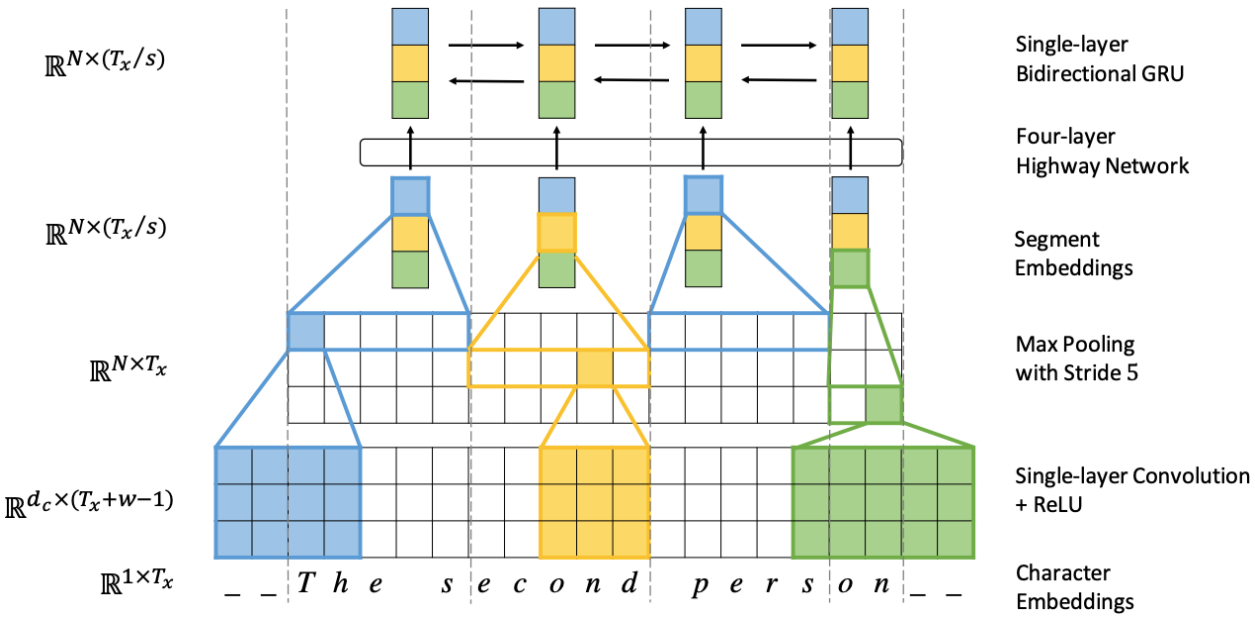
- **Why is BPE-Dropout useful?**
  - currently, most frequent words are **merged** together, so highly frequent words appear **whole** within the vocabulary
  - it is thus up to the **infrequent** words to provide the vocabulary with **morphology** (i.e how to combine morphemes like prefixes, suffixes, common character combinations, etc... to generate words)
  - the problem is that **infrequent** words are **infrequent**, so the model will have less opportunities to learn how to **compose** together the **subwords** in order to understand **infrequent** words, and thus, **generalise** better
  - with **BPE-Dropout**, we obtain **subwords** from **frequent** words, which will provide a more reliable, consistent signal, vis a vis word composition
  - for instance, in the example above, it finds "un", "relate" and "ed" as **subword** units, all of which are more useful than just having "unrelated"as part of the vocabulary
  - with **BPE-Dropout**, the `BLEU` score is **consistently** increased by 1

1.4 Character-Level Neural Machine Translation

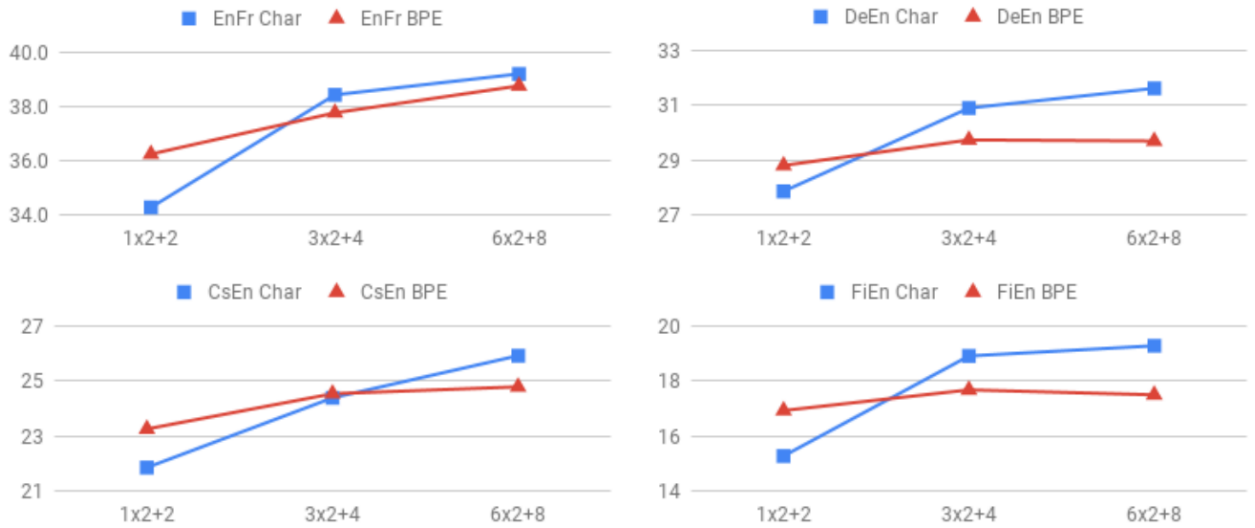
- **What is character-level NMT?**
  - a model learns to translate to and from **sequences of characters**
  - this is in contrast to translating sequences of **words** or **subwords**
- **What are the 3 principal advantages of character-level NMT?**
  1. **Open Vocabulary**: if our vocabulary is composed of characters, it will most likely be **open**

2. **Segmentation**: don't need to think about complicated **heuristics** for subword segmentation (for example, as we saw above, BPE doesn't necessarily produce the most meaningful segmentations, and English is a relatively easy language to segment)
3. **Neural Networks**: NNs are powerful enough to learn directly from character sequences

- **What is the critical disadvantage of character-level NMT?**
  - using **characters** will make input sequences extremely long (in terms of tokens)
  - this significantly slows down **training** and **decoding** (by $\times 2 - 8$ for training alone)
- **In what sense do character-level models affect the understanding of meaning in translated sentences?**
  - when operating with **words** (and even **subwords**), we work with **units** which have **self-contained** meaning
  - this makes it easy to, for example, align these units together
  - there are also clear **semantic dependencies** between these units (either in generating a coherent sentence, or a word)
  - when operating at the **character-level**, this meaningfulness is no longer clear:
    - what inherent meaning does "a" have?
    - how do you even align characters between words from different languages?
    - what sort of dependency do the first and fifth character share?
- **Which approach works best, subword or character level NMT?**
  - generally, operating over **subwords** has lead to better results in translation tasks
  - however, using **characters** is an interesting idea which has received attention over recent years
- **What approaches to character-level NMT have been used?**
  1. **Convolutions**
     - in Fully Character-Level Neural Machine Translation without Explicit Segmentation they use **convolutions** to reduce the sequence length
     - an input sequence of characters is converted into a matrix, with columns as the **character embeddings**
     - a set of different **convolutional filters** are applied across the matrix
     - **max pooling** and different **strides** are used to generate **segment embeddings**: embeddings which represent a segment of the character sequence (each segment embedding might represent a group of characters)
     - these **segment embeddings**, unlike segmentations discussed above, can be **overlapping** (i.e 2 different segments might represent the same character)
     - finally, the segment embeddings are passed through a **highway network** and a **bidirectional GRU** (with **attention**) to generate translations



2. **Deep LSTms**
   - in Revisiting Character-Based Neural Machine Translation with Capacity and Compression, they train very deep **attentional LSTM encoder-decoders** (also using convolutions, as above)
   - for deep models, the **character-level** models performs better (although takes ×8 to train)
   - for shallower models, BPE is best



. 4: For each translation task, as the character-level model gets deeper, it outperforms the subword-leevel model, when evaluated on `BLEU`.

- **What other alternatives have been proposed, beyond character-level segmentations?**

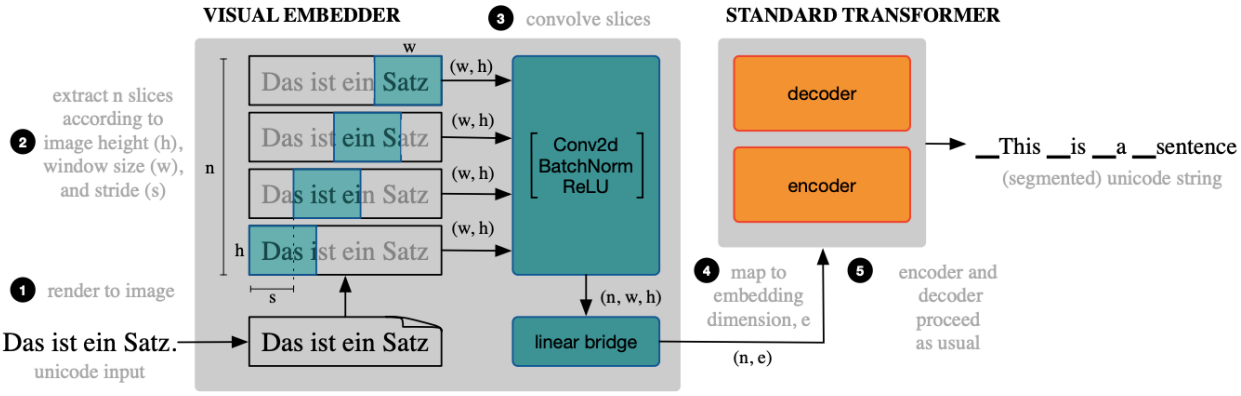
1. **Byte-Level**
   - in ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models they convert **characters** into **byte strings**, according to their **unicode** code

- for example, “like” would be represented by the byte sequence:

$$\texttt{0x6C}, \texttt{0x69}, \texttt{0x6B}, \texttt{0x65}$$

- using this **byte-level** tokenisation means that `ByT5` can handle a lot more languages/scripts, and was shwon to be more **robust** to noise
- however:
  * there are a lot of **unicode** characters, so a **large vocabulary** size, and thus, longer trainig
  * non-ASCII characters have a longer byte-sequence, which can make them more **expensive** to model, thus making the model potentially unfair (it’ll be biased towards ASCII-based languages)

2. **Pixel-Level**
   - in Robust OpenVocabulary Translation from Visual Text Representations, they propose using **images**, and translate based on **pixel information**
   - they process the text in images using different segments at a time (i.e using a sliding window), which get converted into embeddings using **convolutional layers**; these embeddings then get procesed as if they were standard text embeddings

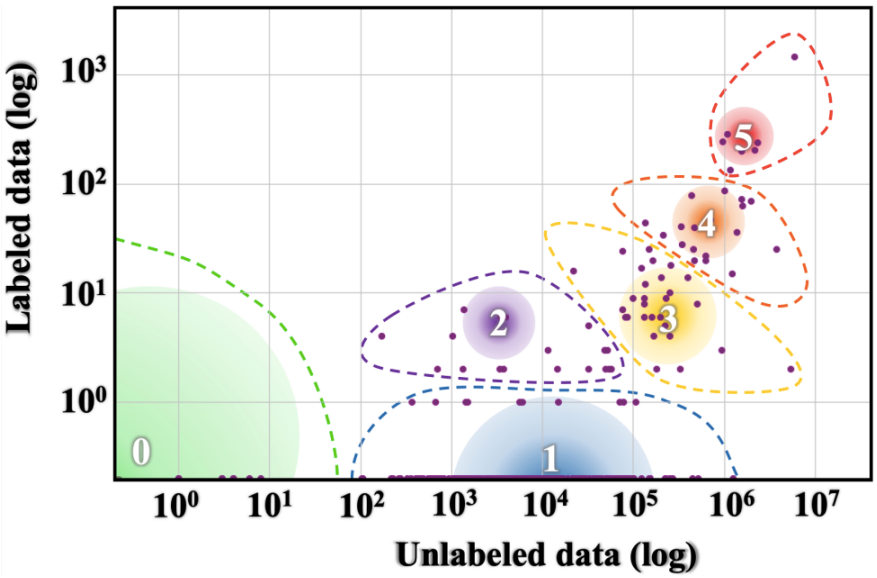


- this approach is more **robust**:
  * it better handles **misspellings** (for instance, “langauge”instead of “langauge since the 2 are visually similar)
  * picks up on **visual similarities** (i.e better handles rare Chinese characters)

2 Low Resource Machine Translation

2.1 Low Resource Languages

- **What is a low-resource language?**
  - a **multifactored** problem
  - certain **languages** are **rarely** used in NLP tasks, due to lack of availability
  - this is caused by a variety of factors, such as lack of **resources** (intellectual, economical, political) from the country of origin
  - this means that these languages are rarely available for models to learn



. 5: Data for 500 languages. The y-axis counts the number of **annotated resources** available for a particular language (in a given dataset full of annotated data). The x-axis counts the number of **unannotated resources** available for a particular langugage (based on the number of wikipedia articles in a given language). The authors split languages into 5 classes. Class 0 are those languages which are "left behind": there is barely little to no annotated data on them, and less than 100 unannotated sources for each of them. On the other hand, class 5 are the "winners": languages from countries with a lot of social, economic and political influence, countries with enough resources to fund language research, countries with powerful universities, etc ...

Class	5 Example Languages	#Langs	#Speakers	% of Total Langs
0	Dahalo, Warlpiri, Popoloca, Wallisian, Bora	2191	1.2B	88.38%
1	Cherokee, Fijian, Greenlandic, Bhojpuri, Navajo	222	30M	5.49%
2	Zulu, Konkani, Lao, Maltese, Irish	19	5.7M	0.36%
3	Indonesian, Ukranian, Cebuano, Afrikaans, Hebrew	28	1.8B	4.42%
4	Russian, Hungarian, Vietnamese, Dutch, Korean	18	2.2B	1.07%
5	English, Spanish, German, Japanese, French	7	2.5B	0.28%

. 6: Distribution of languges between classes. We can see that the underrepresented languages form a majority of all languages explored here.

2.2 Creating Datasets

2.2.1 Web Crawling on Translated Websites

- **What is the easiest way of obtaining high-quality, parallel, multilingual data from the Internet?**
  - a variety of websites have versions in **multiple languages** (for example, https://www.un.org/en/ is the UN website in English, whilst https://www.un.org/zh/ is the UN website in Chinese)
  - determining this is fairly straightforward from the URL alone, or looking into website metadata
  - thus, a page with the **same information** can be found in multiple languages

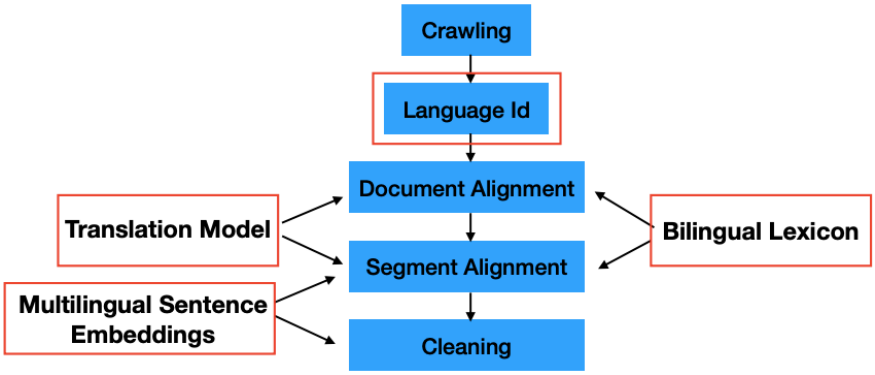
- tags/attributes can be used to **align** sentences together, which after some deduplication and filtering, gives high-quality parallel, multilingual data

- **What is the main issue with this approach?**
  - such websites are not too common
  - even the ones that have translations available, probably only have entries in a small subset of well-known languages (which we already have enough data for)

2.2.2 Web Crawling on Monolingual Websites

- **What alternatives are there to using translated websites?**
  - the Internet has **zettabytes** ($10^{12}GB$) worth of **monolingual** data
  - there have been many attempts to **crawl** and access all this data, such as with **Common Crawl** or **Internet Archieve**
  - this has resulted in crawling **petabytes** ($10^{6}GB$) of **monolingual data**

- **How can monolingual data be used to generate parallel, multilingual data?**
  1. Identify the **language** of the page
  2. **Align** pages (documents) together. This isn't done for all sites found, but rather by using heuristics:
     - common URLs
     - using a translation dictionary and seeing overlap between pages
  3. Within **aligned** documents, try **aligning** sentences. Again, use heuristics:
     - sentences with very different lengths are likely not translations
     - use **translation models** to find possible matches
     - project sentences into **common embedding space** (i.e `LASER`) and use **nearest neighbours** to find parallel sentences
  4. **Filter** sentence alignments to ensure we keep highest quality translations



. 7: Generating parallel data from monolingual sites. Steps highlighted in red are those that might be difficult for low resource languages.

- **Why is this approach not optimal for low resource languages?**
  - in the steps outlined above, we need **robust** language information: translation dictionaries, translation models, etc ...

- we have really good tools, which we have tried and tested multiple times, for common languages like English or Spanish
- however, with **low resource** languages, these aren't widely available or as robust
- thus, the above technique are **error prone**, and can lead to a large amount of **false positive** alignments

	Parallel		
	CCAligned	**ParaCrawl v7.1**	**WikiMatrix**
#languages	137	41	85
Source	CC 2013–2020	selected websites	Wikipedia
Filtering level	document	sentence	sentence
Langid	FastText	CLD2	FastText
Alignment	LASER	Vec/Hun/BLEU-Align	LASER
Evaluation	TED-6	WMT-5	TED-45

. 8: Examples of aligned datasets based on monolingual data.

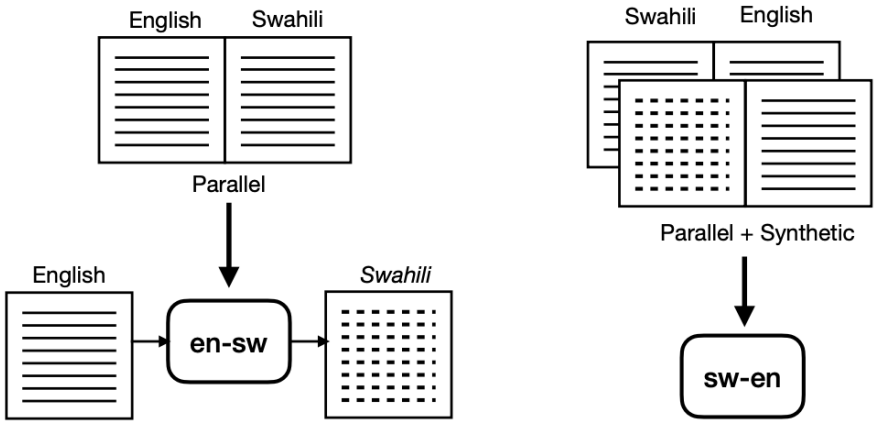
		Parallel		
		CCAligned	**ParaCrawl v7.1**	**WikiMatrix**
#langs audited / total		65 / 119	21 / 38	20 / 78
%langs audited		54.62%	55.26%	25.64%
#sents audited / total		8037 / 907M	2214 / 521M	1997 / 95M
%sents audited		0.00089%	0.00043%	0.00211%
macro	C	29.25%	76.14%	23.74%
	X	29.46%	19.17%	68.18%
	WL	9.44%	3.43%	6.08%
	NL	31.42%	1.13%	1.60%
	offensive	0.01%	0.00%	0.00%
	porn	5.30%	0.63%	0.00%

. 9: Native speakers were asked to evaluate the parallel data generated within these datasets. `C` denotes the percentage of sentence pairs which were correct. `X` denotes the percentage of setnence pairs which weren't correct. `WL` denotes the percentage of sentence pairs which had a wrong translation language. `NL` denotes the percentage of sentence pairs which were aligned with text which didn't conform a language. As can be seen, `CCAligned` and `WikiMatrix` obtain fairly poor results. What is most worrisome is that these datasets were used for training low resource models for quite a few years, despite the low data quality.

2.3 Back Translation

- **What is the aim of back translation?**
  - **back translation** was suggested in Improving Neural Machine Translation Models with Monolingual Data
  - say we have a **source** language S, and a **target** language T
  - we have a large amount of **monolingual** data for S,T, and just a few examples of parallel data between S and T

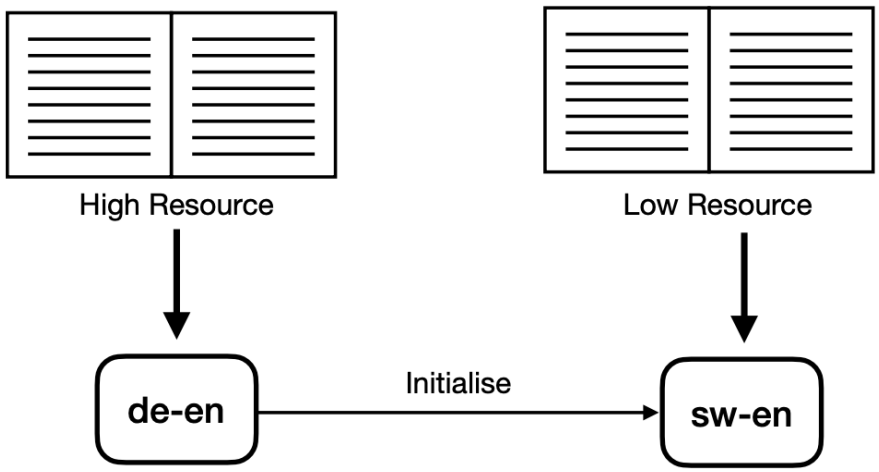
- then, with back translation:
  1. Using the **parallel** data, train a simple translation model $\mathcal{T}$, which translates $T \rightarrow S$
  2. Using $\mathcal{T}$, generate **synthetic training samples**, by converting the monolingual data in T
  3. We then train a **back translation model** from $S \rightarrow T$, by using both the synthetic parallel data, alongside the high quality parallel data
  4. This can be iterated to progressively improve the translation quality
- when tested, **back translation models** obtained better performance than simpler **phrase-based models**
- this allows training **robust** models by using just a few parallel training samples and a lot of **monolingual data**
- having high quality parallel data is still important: the **back translation model** might fail if the syntehtic-pair generation system is too weak



2.4 Transfer Learning

2.4.1 Parallel Data

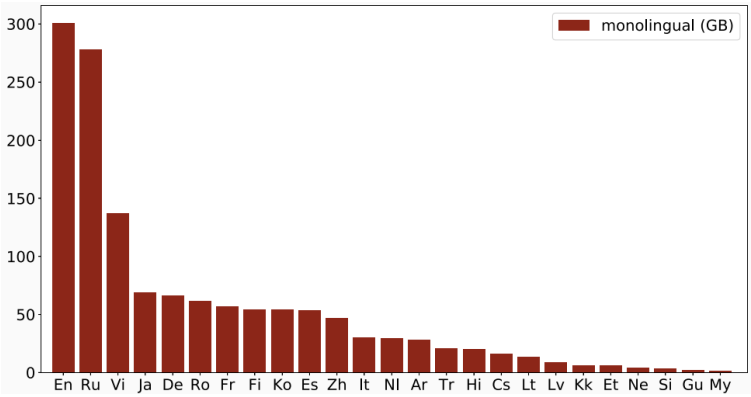
- **How can transfer learning be used when parallel data is available?**
  - we can **pretrain** a model on **high resource** languages, with the desired **target language**
  - we fine-tune by using the **low resource** parallel data



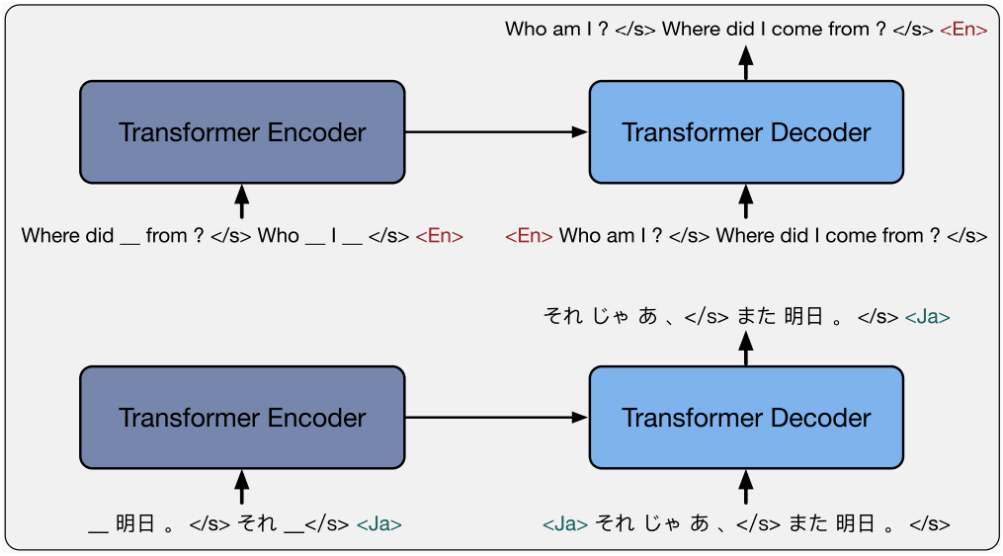
- **What factors affect the quality of translation when using transfer learning?**
  - initially (Transfer Learning for Low-Resource Neural Machine Translation) showed that this worked well for **low resource** Turkic languages
  - in Trivial Transfer Learning for Low-Resource Neural Machine Translation they showcase how in fact, source and target don't need to be linguistically related
  - in Choosing Transfer Languages for Cross-Lingual Learning, they extensively investigated the effect of pretraining languages, and determined that the most important factors were:
    * data size
    * quality of pretrained model
    * lexical overlap between subwords

2.4.2 mBART

- **Why were large pretrained models not used for machine translation before 2020?**
  1. **Architecture**: large pretrained models like `BERT` are **encoders** (i.e good at encoding embeddings, not so good at generating), whereas MT requires an **encoder-decoder** architecture
  2. **Time**: pretrained models require **many** parameters, making them prohibitively expensive for **training** and **decoding**
- **What is `mBART`?**
  - `mBART` was introduced in Multilingual Denoising Pre-training for Neural Machine Translation
  - a **pretrained** model using a **monolingual corpus**, containing 25 languages from the **Common Crawl**

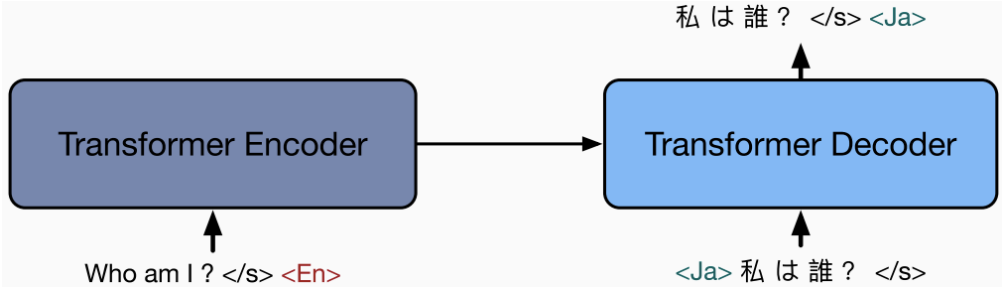


- **How was `mBART` pretrained?**
  - pretrain on all the multilingual data on **denoising** tasks:
    * **Masked Token Prediction**: given an input sentence, predict the token that would appear in place of a mask
    * **Sentence Order**: given 2 sentences, determine in which order they appear
  - it also had to **predict** the text language



  - pretraining took **weeks**, over 100s of GPUs

- **How was `mBART` fine-tuned?**
  - for **fine-tuning**, the parallel data was used for translation



  - notice, the resulting model learns to represent **all** languages at once

- **How does `mBART` perform on high/low-resource languages?**
  - `mBART` was compared with both **high resource** and **low resource** languages
  - as a **baseline**, they considered a **randomly initialised** `mBART` model

Languages	En-Gu		En-Kk		En-Vi		En-Tr		En-Ja		En-Ko	
Data Source	WMT19		WMT19		IWSLT15		WMT17		IWSLT17		IWSLT17	
Size	10K		91K		133K		207K		223K		230K	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
Random	0.0	0.0	0.8	0.2	23.6	24.8	12.2	9.5	10.4	12.3	15.3	16.3
mBART25	**0.3**	**0.1**	**7.4**	**2.5**	**36.1**	**35.4**	**22.5**	**17.8**	**19.1**	**19.4**	**24.6**	**22.6**

Languages	En-Nl		En-Ar		En-It		En-My		En-Ne		En-Ro	
Data Source	IWSLT17		IWSLT17		IWSLT17		WAT19		FLoRes		WMT16	
Size	237K		250K		250K		259K		564K		608K	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
Random	34.6	29.3	27.5	16.9	31.7	28.0	23.3	34.9	7.6	4.3	34.0	34.3
mBART25	**43.3**	**34.8**	**37.6**	**21.6**	**39.8**	**34.0**	**28.3**	**36.9**	**14.5**	**7.4**	**37.8**	**37.7**

Languages	En-Si		En-Hi		En-Et		En-Lt		En-Fi		En-Lv	
Data Source	FLoRes		ITTB		WMT18		WMT19		WMT17		WMT17	
Size	647K		1.56M		1.94M		2.11M		2.66M		4.50M	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
Random	7.2	1.2	10.9	14.2	22.6	17.9	18.1	12.1	21.8	20.2	15.6	12.9
mBART25	**13.7**	**3.3**	**23.5**	**20.8**	**27.8**	**21.4**	**22.4**	**15.3**	**28.5**	**22.4**	**19.3**	**15.9**

Languages	Cs	Es	Zh	De	Ru	Fr
Size	11M	15M	25M	28M	29M	41M
Random	16.5	33.2	**35.0**	**30.9**	**31.5**	**41.4**
mBART25	**18.0**	**34.0**	33.3	30.5	31.3	41.0

  - for all **low resource** languages, **pretraining** `mBART` leads to a significant performance improvement
  - for **high resource** languages, pretraining doesn't benefit performance significantly

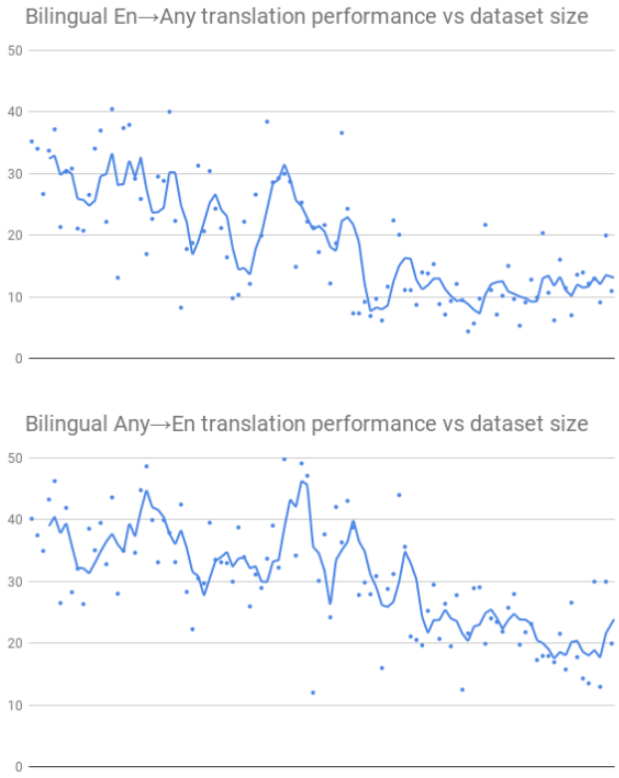
- **How has `mBART` been extended?**
  - developed `mBART50` (Multilingual Translation with Extensible Multilingual Pretraining and Finetuning):
    1. **Languages**: can handle 50 different languages
    2. **Many-to-Many Translation**: new fine-tuning allows for many-to-many translation
  - used as a **baseline** for most **low resource** translation tasks

2.4.3 Multilingual Models

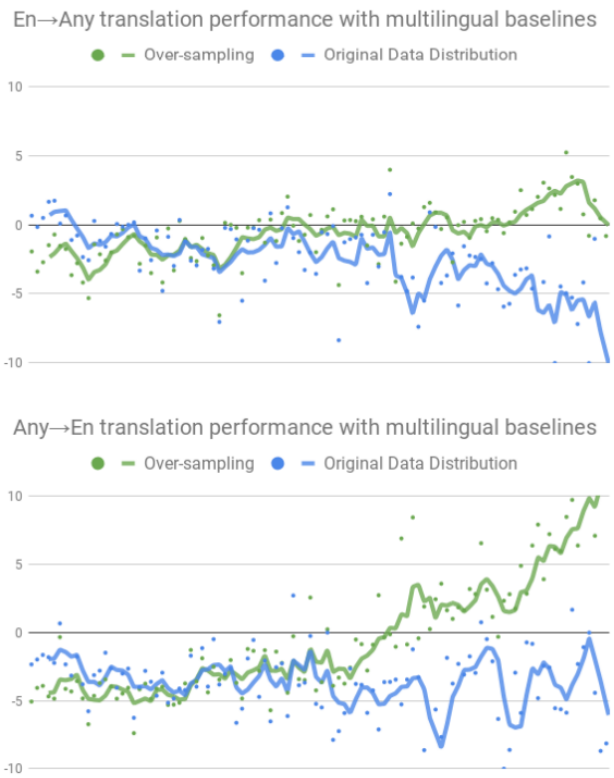
- **What are multilingual models?**
  - currently, we've mainly operated using many-to-1 (i.e bunch of languages to English) or 1-to-many models (i.e English to bunch of languages)
  - **multilingual models** seek to define a **many-to-many** translation paradigm (i.e from Uzbek to Urdu, instead of Uzbek-English, English-Urdu)

- **What approaches have been used for multilingual models?**
  1. An Analysis of Massively Multilingual Neural Machine Translation for Low-Resource Languages: use a small number of **related** languages for many-to-many translation
  2. Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges: use a **large** amount of languages (103)

- **What 2 concepts do multilingual models have to balance?**
  - **Knowledge Transfer**: being capable of translating between so many languages means that the model will learn to better understand the **structure** of language
  - **Knowledge Interference**: needing to tranlsate between languages will degrade performance (need to better generalise; if certain languages have specific features which aren't shared between other languages, there'll be a performance drop)
- **How does the performance of multilingual models vary when used for high/low resource languages?**
  - in Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges they compare how a bilingual model (English-to-many, many-to-English) performs compared to a multilingual model
  - 2 different multilingual models are used:
    * the **original** data distribution, where high resource and low resource data appears in the same proportions
    * an **oversampled** data distribution, where low resource languages are **upsampled**, so that they constitute the same proportion of training data as high resource languages



. 10: `BLEU` scores for the bilingual model. Scores are arranged from left to right, based on language presence in training (low resource languages to the right, high resource languages to the left).



. 11: Difference between BLEU scores for the bilingual and multilingual models. Scores are arranged from left to right, based on language presence in training (low resource languages to the right, high resource languages to the left). In blue, the performance of the multilingual model with the original data distribution; in green, the performance after using upsampling.

- these results showcase that:
  1. **Negative Interference**: seems to significantly affect high resource models, and low resource models with the original data distribution
  2. **Positive Transfer**: seems to significantly benefit low resource languages, particularly during **encoding**, and when using upsampling
- more recently, we've been able to show that even high resource languages benefit from **multilingual** models:

MMT	Model	cs-en	de-en	ha-en	is-en	ja-en	ru-en	zh-en	Avg	Incremental Δ
✗	**Bilingual**	28.9	41.5	15.9	30.3	19.7	40.2	34.8	30.2	—
✗	**+ Backtranslation**	28.3	38.0	28.3	34.5	21.1	38.0	30.8	31.3	+1.1
✗	**+ Finetuning**	30.4	42.8	30.3	35.5	24.6	39.5	36.2	34.2	+2.9
✓	**+ Multilingual**	32.1	43.8	36.1	39.4	26.7	40.6	36.9	36.5	+2.3
✓	**+ Ensemble**	32.3	44.5	37.2	39.9	27.2	40.9	37.8	37.1	+0.6
✓	**+ Reranking**	32.7	44.4	38.2	40.5	27.8	41.4	38.0	37.6	+0.5
✗	**WMT20 Winner**	29.9	43.8	—	—	26.6	39.2	36.9		
	Δ over WMT20	+2.8	+0.6	—	—	+1.2	+2.2	+1.1		

MMT	Model	en-cs	en-de	en-ha	en-is	en-ja	en-ru	en-zh	Avg	Incremental Δ
✗	**Bilingual**	33.1	38.7	14.7	25.8	25.4	25.8	40.0	29.1	—
✗	**+ Backtranslation**	33.1	39.6	23.1	29.4	26.1	25.7	42.4	31.3	+2.3
✗	**+ Finetuning**	35.7	39.5	23.3	29.4	27.7	26.0	43.0	32.1	+0.7
✓	**+ Multilingual**	36.4	40.8	24.6	31.2	29.7	26.8	43.6	33.3	+1.2
✓	**+ Ensemble**	36.8	41.1	25.0	32.5	29.7	26.9	43.6	33.7	+0.4
✓	**+ Reranking**	37.2	41.1	25.5	32.8	29.7	27.4	43.6	33.9	+0.2
✓	**+ Postprocessing**	39.8	42.6	25.5	34.5	29.8	28.8	48.2	35.6	+1.7
✗	**WMT20 Winner**	36.8	38.8	—	—	28.4	25.5	47.3		
	Δ over WMT20	+3.0	+3.8	—	—	+1.4	+3.3	+0.9		

. 12: The largest performance improvement from the **bilingual baseline** is in low resource languages, like czech, hausa or icelandinc. However, even high resource languages, like chinese or russian benefit from the multilingual model. Results from Facebook AI's WMT21 News Translation Task Submission.

2.5 Evaluating Low Resource Machine Translation

- **Why is automatic evaluation of low resource MT systems particularly challenging?**
  - in short, because most automatic evaluation systems (like `BLEU`) are mainly designed for high resource languages
  - as such, it is harder for them to intuitively evaluate low resource languages (i.e which can be more **morphologically** rich)
  - how comparable are `BLEU` scores for low resource languages with scores for high resource languages?
- **What is the best way of evaluating low resource translations?**
  - currently, the best way is **human evaluation**
  - for this, researchers need to be able to better connect with different **language communities**

3 NLP and Ethics