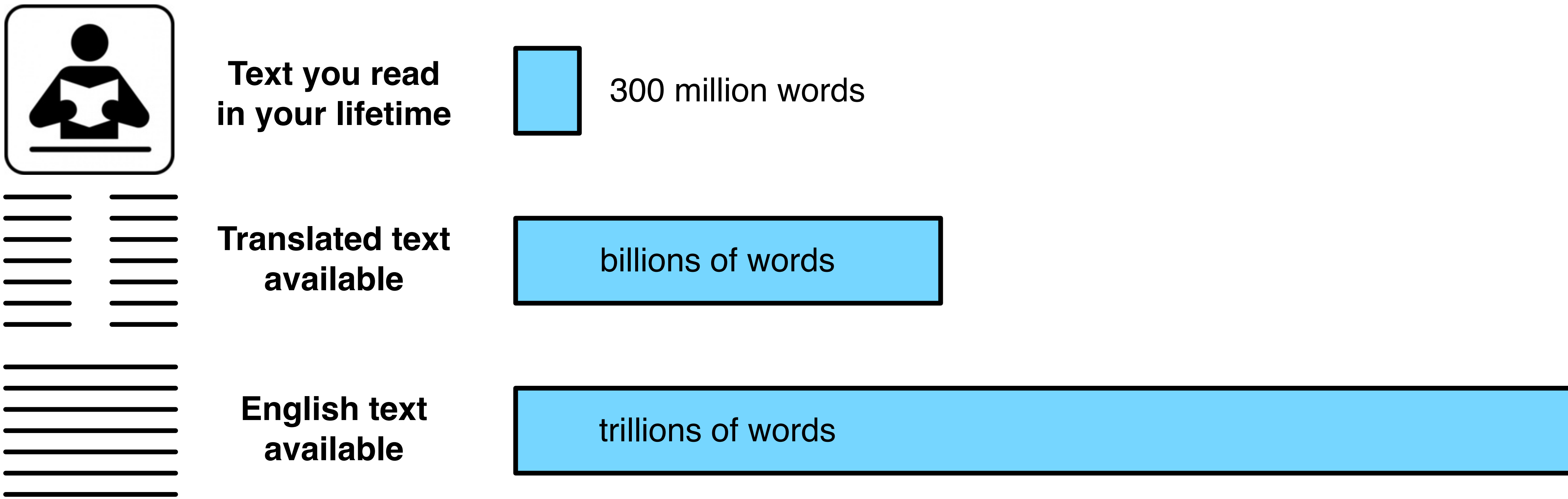
Corpus Acquisition from the Internet

Philipp Koehn
partially based on slides from Christian Buck

8 November 2022

Big Data

For many language pairs, lots of text available.



Mining the Web

- Largest source for text: the World Wide Web

Common Crawl

  - publicly available crawl of the web
  - hosted by Amazon Web Services, but can be downloaded
  - regularly updated (semi-annual)
  - 2-4 billion web pages per crawl

- Currently filling up hard drives in our lab

Monolingual Data

- Starting point: 35TB of text

- Processing pipeline [Buck et al., 2014]
  - language detection
  - deduplication
  - normalization of Unicode characters
  - sentence splitting

- Obtained corpora

Language	Lines (B)	Tokens (B)	Bytes	BLEU (WMT)
English	59.13	975.63	5.14 TB	-
German	3.87	51.93	317.46 GB	+0.5
Spanish	3.50	62.21	337.16 GB	-
French	3.04	49.31	273.96 GB	+0.6
Russian	1.79	21.41	220.62 GB	+1.2
Czech	0.47	5.79	34.67 GB	+0.6

Parallel Data

- Basic processing pipeline [Smith et al., 2013]
  - find parallel web pages (based on URL only)
  - align document by HTML structure
  - sentence splitting and tokenization
  - sentence alignment
  - filtering (remove boilerplate)

- Obtained corpora

	French	German	Spanish	Russian	Japanese	Chinese
Segments	10.2M	7.50M	5.67M	3.58M	1.70M	1.42M
Foreign Tokens	128M	79.9M	71.5M	34.7M	9.91M	8.14M
English Tokens	118M	87.5M	67.6M	36.7M	19.1M	14.8M
	Bengali	Farsi	Telugu	Somali	Kannada	Pashto
Segments	59.9K	44.2K	50.6K	52.6K	34.5K	28.0K
Foreign Tokens	573K	477K	336K	318K	305K	208K
English Tokens	537K	459K	358K	325K	297K	218K

- Much more work needed!

Data Cleaning and Subsampling

- Not all data useful – some may be harmful
- Removing data based on
  - domain relevance
  - alignment quality
  - redundancy
  - bad language (orthography, non-words)
  - machine translated or poorly translated
- Removing bad data always reduces training time
- Removing bad data sometimes helps quality
- Clean data approach (only using high quality data) helps in limited domains

corpus crawling

Finding Monolingual Text

- Simple Idea
  1. Download many websites
  2. Extract text from HTML
  3. Guess language of text
  4. Add to corpus
  5. Profit
- Turns out all these steps are quite involved

Common Crawl

- Non-profit organization
- Data
  - publicly available on Amazon S3
  - e.g. January 2015: 140TB / 1.8B pages
- Crawler
  - Apache Nutch
  - collecting pre-defined list of URLs

extracting text

A Web Page


ABOUT CONTACT US FORUMS HOME LINUX HOW-TO & TUTORIALS SHELL SCRIPTS
RSS/FEED
nixCraft
INSIGHT INTO LINUX ADMIN WORK
Bash Shell: Find Out Linux / FreeBSD / UNIX System Load Average
by NIXCRAFT on MARCH 23, 2005 · 8 COMMENTS· LAST UPDATED AUGUST 8, 2013
in LINUX, MONITORING, SYS ADMIN
Yes, I know we can use the uptime command to find out the system load average. The uptime command displays the current time, the length of time the system has been up, the number of users, and the load average of the system over the last 1, 5, and 15 minutes. However, if you try to use the uptime command in script, you know how difficult it is to get correct load average. As the time since the last, reboot moves from minutes, to hours, and an even day after system rebooted. Just type the uptime command:
$ uptime
Google™ Custom Search
nixCraft
Follow
+1
+ 137,320
LATEST LINUX/UNIX Q & A
How To Patch and Protect OpenSSL Vulnerability # CVE-2015-0291 CVE-2015-0204 [19/March/2015]

HTML Source


```
load-average.html#comments" rel="nofollow">8 comments</a></span>&middot; <span>LAST UPDATED
<abbr
class="updated" title="2013-08-08">August 8, 2013</abbr><span><p
class='headline_meta'> in <span><a
rel='tag' href='http://www.cyberciti.biz/tips/category/linux'>Linux</a>, <a
rel='tag' href='http://www.cyberciti.biz/tips/category/monitoring'>Monitoring</a>, <a
rel='tag' href='http://www.cyberciti.biz/tips/category/sys-admin'>Sys admin</a></span>
</p></div><div
class="format_text entry-content"><p><span
class="drop_cap">Y</span>es, I know we can use the <kbd>uptime</kbd> command to find out the
system load average. The uptime command displays the current time, the length of time the
system has been up, the number of users, and the load average of the system over the last 1,
5, and 15 minutes. However, if you try to use the uptime command in script, you know how
difficult it is to get correct load average. As the time since the last, reboot moves from
minutes, to hours, and an even day after system rebooted. Just type the uptime command:<br
/> <span
id="more-631"></span><br
/> <code>$ uptime</code><br
/> Sample outputs:</p><pre>1:09:01  up 29 min,  1 user,  load average: 0.00, 0.00, 0.00</pre>
<p>OR<br
/> <code>$ uptime</code><br
/> Sample outputs:</p><pre>2:13AM  up 34 days, 16:15, 36 users, load averages: 1.56, 1.89,
2.06</pre><p>Traditionally, UNIX administrators used sed and other shell command in scripting
to get correct value of load average. Here is my own modified hack to save the time<br
/> <code>$ uptime | awk -F'load averages:' '{ print $2 }'</code><br
/> OR better use the following code:<br
/> <code>$ uptime | awk -F'[a-z]:' '{ print $2}'</code><br
/> Output taken from my <strong>OS X desktop</strong>:</p><pre> 1.24 1.34 1.35</pre><p>Output
taken from my <strong>Ubuntu</strong> Linux server:</p><pre> 0.00, 0.01, 0.05</pre><p>Output
taken from my <strong>RHEL</strong> based server:</p><pre> 0.24, 0.27, 0.21</pre><p>Output
taken from my <strong>FreeBSD</strong> based server:</p><pre> 0.71, 0.71, 0.58</pre><p>Please
note that command works on all variant of UNIX operating systems.</p><h2>See also</h2>
<ul><li>See <a
href="http://bash.cyberciti.biz/monitoring/chksysload.bash.php">chksysload.bash</a> script to
```

Method 1: Strip Tags

```
LAST UPDATED August 8, 2013 in Linux , Monitoring , Sys
admin Y es, I know we can use the uptime command to find
out the system load average. The uptime command displays
the current time, the length of time the system has been
up, the number of users, and the load average of the
system over the last 1, 5, and 15 minutes. However, if
you try to use the uptime command in script, you know how
difficult it is to get correct load average. As the time
since the last, reboot moves from minutes, to hours, and
an even day after system rebooted. Just type the uptime
command: $ uptime Sample outputs: 1:09:01 up 29 min, 1
user, load average: 0.00, 0.00, 0.00
```

Method 2: HTML Parser

LAST UPDATED August 8, 2013

in Linux, Monitoring, Sys admin

Y

es, I know we can use the uptime command to find out the system load average. The uptime command displays the current time, the length of time the system has been up, the number of users, and the load average of the system over the last 1, 5, and 15 minutes. However, if you try to use the uptime command in script, you know how difficult it is to get correct load average. As the time since the last, reboot moves from minutes, to hours, and an even day after system rebooted. Just type the uptime command:

$ uptime

Sample outputs: 1:09:01 up 29 min, 1 user, load average: 0.00, 0.00, 0.00

language detection

What Language?

Muitas intervenções alertaram para o facto de a política dos sucessivos governos PS, PSD e CDS, com cortes no financiamento das instituições do Ensino Superior e com a progressiva desresponsabilização do Estado das suas funções, ter conduzido a uma realidade de destruição da qualidade do Ensino Superior público.

Clues: Letter N-Grams

Muitas intervenções alertaram para o facto de a política dos sucessivos governos PS, PSD e CDS, com cortes no financiamento das instituições do Ensino Superior e com a progressiva desresponsabilização do Estado das suas funções, ter conduzido a uma realidade de destruição da qualidade do Ensino Superior público.

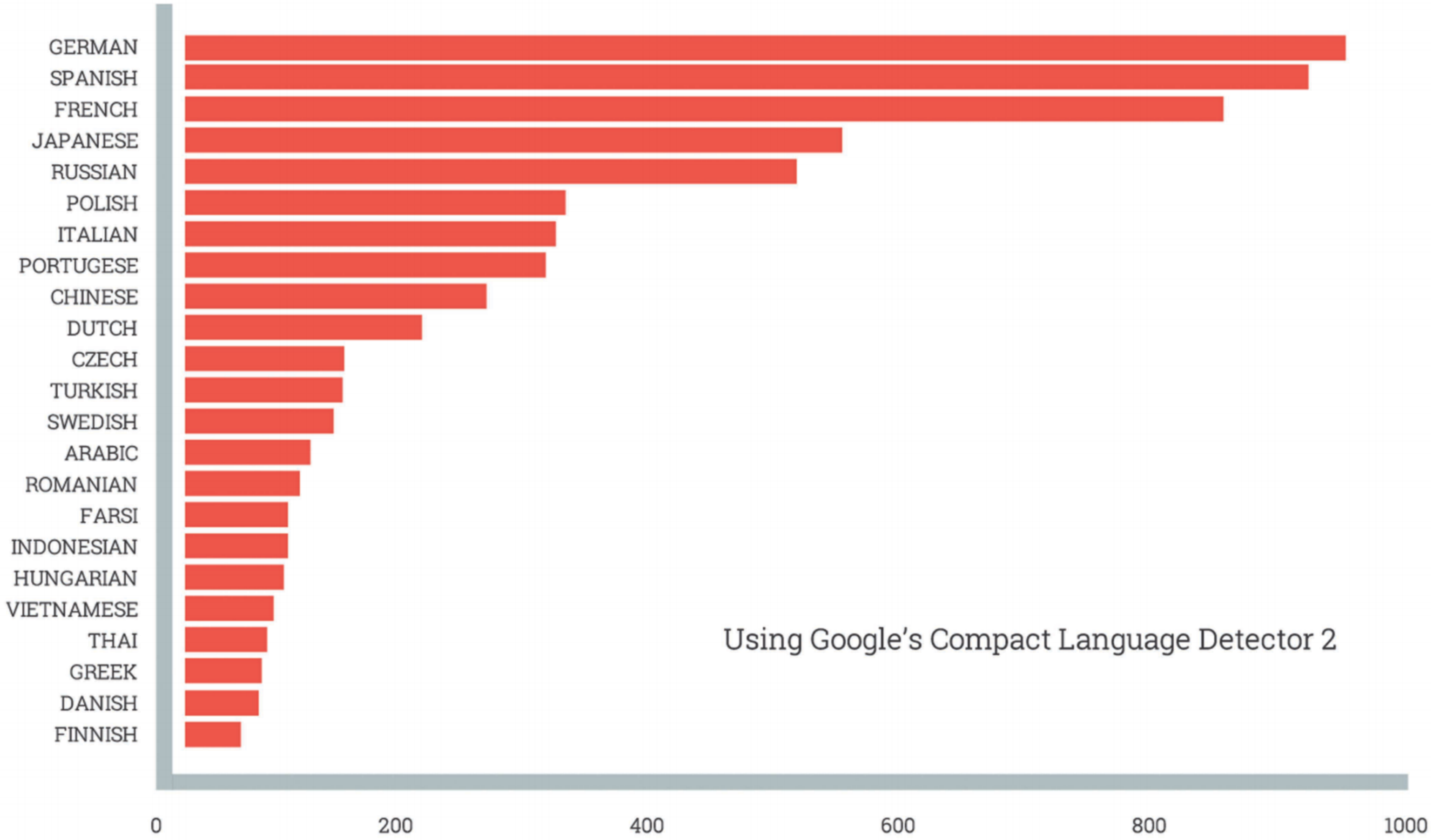
Example: `langid.py`

- Muitas intervenções alertaram
  - prediction: Portuguese
  - high confidence (-90.8)
- Muitas intervenções
  - prediction: Portuguese
  - fairly high confidence (-68.2)
- Muitas
  - prediction: English
  - low confidence (9.1)

Language Identification Tools

- `langid.py` (Lui & Baldwin, ACL 2012)
  - 1-4 grams, NaiveBayes, Feature Selection
- TextCat (based on Cavnar & Trenkle, 1994)
  - similar to langid.py
  - no Feature Selection
- Compact/Chromium Language Detector 2 (Google)
  - takes hints from tld, meta data
  - super fast
  - detects spans of text

Detected Languages in CommonCrawl



(Buck and Heafield, LREC2014)

Most Common English Phrases

Count (M)	Line
1374.44	Add to
816.33	Share
711.68	Unblock User
68.31	Sign in or sign up now!
61.26	Log in
54.77	Privacy Policy
45.18	April 2010
34.35	Load more suggestions
19.84	Buy It Now | Add to watch list
16.64	Powered by WordPress.com

Benefit of Huge Language Models

	BLEU			
	2012	Δ	2013	Δ
Baseline	35.8		30.9	
+ 50M lines	36.3	0.5	31.5	0.6
+ 100M lines	36.5	0.7	31.5	0.6
+ 200M lines	36.6	0.8	31.8	0.9
+ 400M lines	37.0	1.2	31.8	0.9
+ 800M lines	37.3	1.6	31.8	0.9
+ 1.3B lines	37.7	1.9	32.0	1.1

bilingual corpus crawling

Mining Bilingual Text

- Bilingual text = same text in different languages
- Usually: one side translation of the other
- Full page or interface/content only
- Potentially translation on same page
  e.g., Twitter, Facebook posts

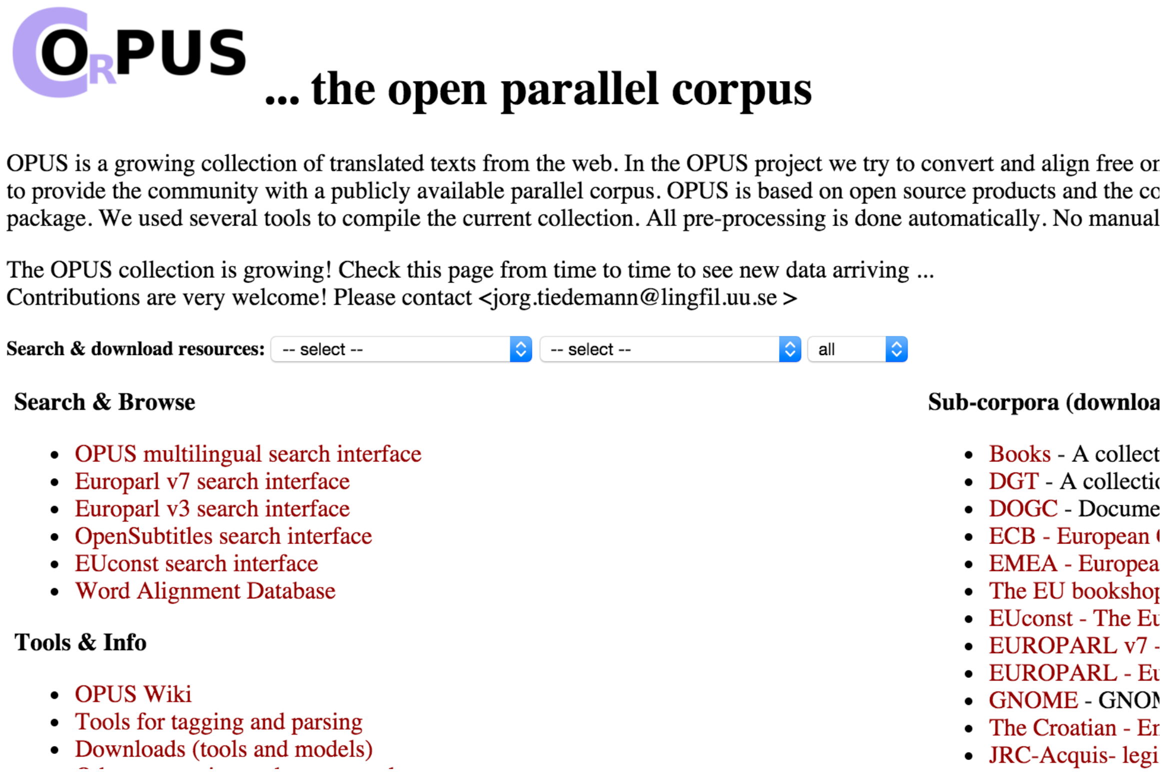
Pipeline

1. Identify web sites worth crawling
2. Crawl web site
3. Language detection — as before
4. Extract text from HTML — as before
5. Align documents
6. Align sentences
7. Clean corpus

identify web sites

Targeted Crawling

- A few web sites with a lot of parallel text, e.g.,
  - European Union, e.g., proceedings of the European Parliament
  - Canadian Hansards
  - United Nations
  - Project Syndicate
  - TED Talks
  - Movie / TV show subtitles
  - Global Voices
- Hand-written tools
  - crawling
  - text extraction
  - document alignment
- Few days effort per site



OPUS ... the open parallel corpus

OPUS is a growing collection of translated texts from the web. In the OPUS project we try to convert and align free or
to provide the community with a publicly available parallel corpus. OPUS is based on open source products and the cc
package. We used several tools to compile the current collection. All pre-processing is done automatically. No manual

The OPUS collection is growing! Check this page from time to time to see new data arriving ...
Contributions are very welcome! Please contact <jorg.tiedemann@lingfil.uu.se >

Search & download resources: -- select -- -- select -- all

Search & Browse

- OPUS multilingual search interface
- Europarl v7 search interface
- Europarl v3 search interface
- OpenSubtitles search interface
- EUconst search interface
- Word Alignment Database

Tools & Info

- OPUS Wiki
- Tools for tagging and parsing
- Downloads (tools and models)

Sub-corpora (downloa

- Books - A collect
- DGT - A collecti
- DOGC - Docume
- ECB - European
- EMEA - Europea
- The EU bookshop
- EUconst - The Eu
- EUROPARL v7 -
- EUROPARL - Eu
- GNOME - GNON
- The Croatian - Er
- JRC-Acquis- legi

Broad Crawling

- Identify many web sites to crawl
  - has the phrase This page in English or variants
  - has link to language flag
  - known to have content in multiple languages (from CommonCrawl)
- Follow links
  - up to n links deep into site
  - up to n links in total
  - only follow links to web pages, not images, etc.
- Avoid crawling sites too deeply that do not have parallel text?
  (requires quick feedback from downstream processing)

document alignment

Document Alignment

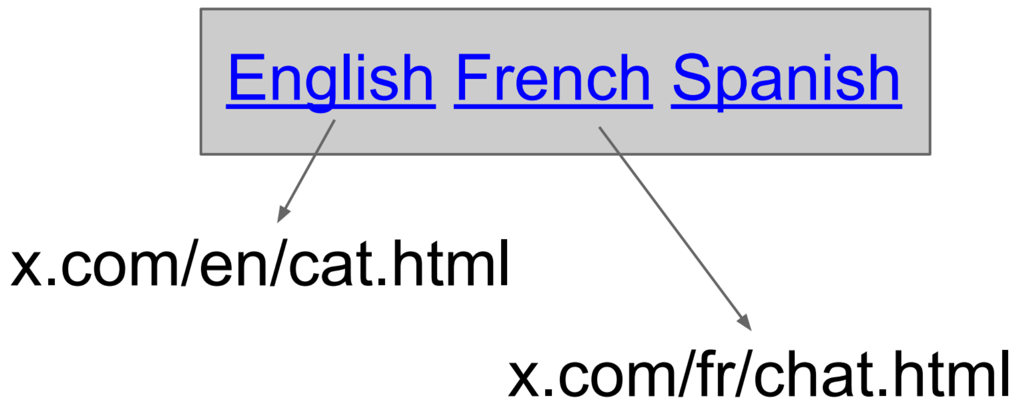
- Early Work: STRAND (Resnik 1998, 1999)
  (Structural Translation Recognition, Acquiring Natural Data)

- Pipeline

  1. candidate generation
  2. candidate ranking
  3. filtering
  4. optional: sentence alignment
  5. evaluation

Link Structure

- Parent page: a page that links to different language versions

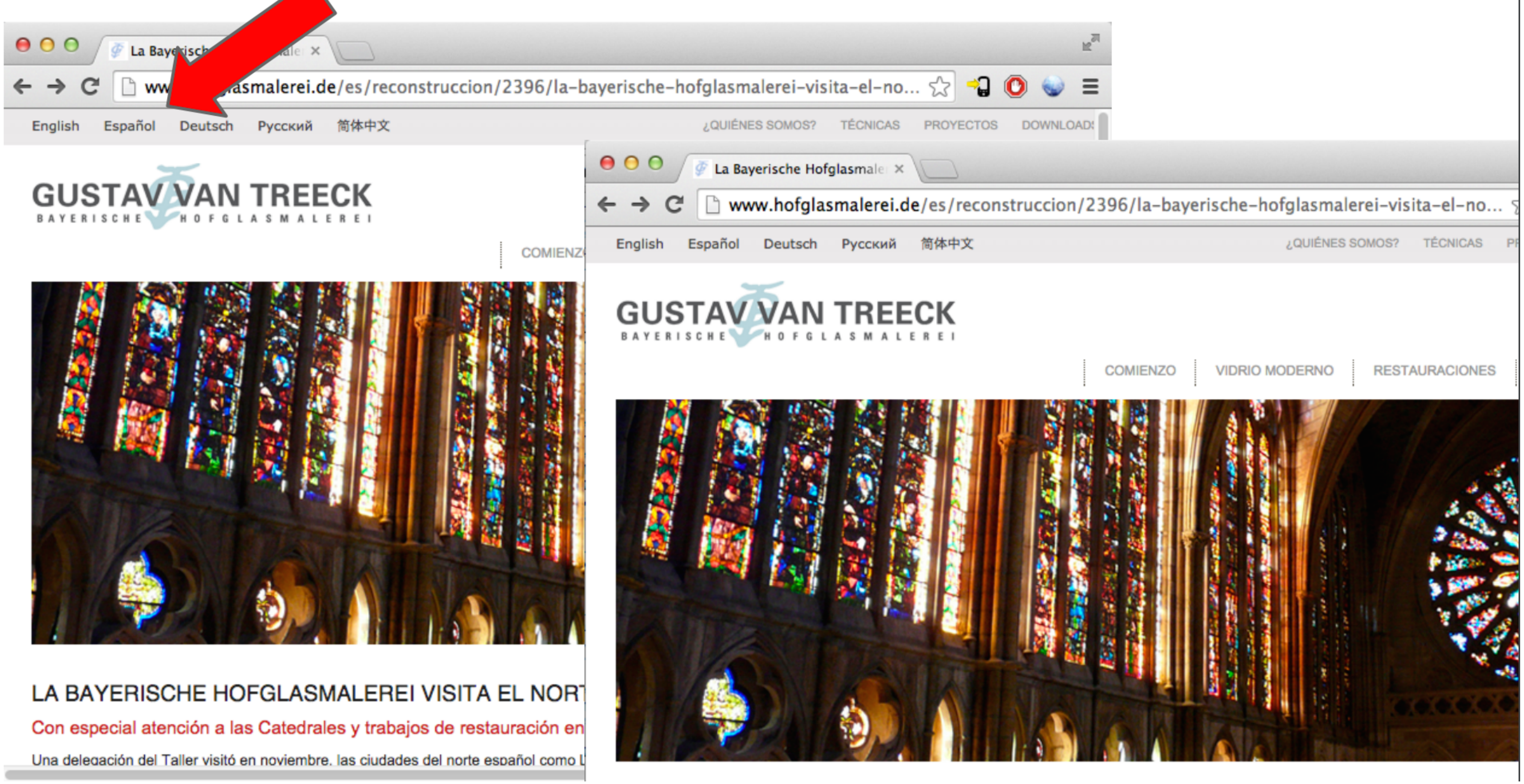


Parent Page Example

Sibling Page

- A page that links to its translation in another language

[redacted] [redacted]



URL Matching

- Often URLs differ only slightly, often indicating language

xyz.com/en/	xyz.com/fr/
xyz.com/bla.htm	xyz.com/bla.htm?lang=FR
xyz.com/the_cat	xyz.fr/le_chat

Finding URL Patterns

- URLs with pattern =en

Count	Pattern
545875	lang=en
140420	lng=en
126434	LANG=en
110639	hl=en
99065	language=en
81471	tlng=en
56968	l=en
47504	locale=en
33656	langue=en
33503	lang=eng
19421	uil=English
15170	ln=en
14242	Language=EN
13948	lang=EN
12108	language=english
11997	lang=engcro
11646	store=en

Finding URL Patterns

- URLs with pattern `lang.*=.*`

Count	Pattern
13948	`lang=EN`
13456	`language=ca`
13098	`switchlang=1`
12960	`language=zh`
12890	`lang=Spanish`
12471	`lang=th`
12266	`langBox=US`
12108	`language=english`
12003	`lang=cz`
11997	`lang=engcro`
11635	`lang=sl`
11578	`lang=d`
11474	`lang=lv`
11376	`lang=NL`
11349	`lang=croeng`
11244	`lang=English`

Document Length

- Extract texts and compare lengths (Smith 2001)

$$\text{Length}(E) \approx C * \text{Length}(F)$$

learned,
language-specific parameter

- Document or sentence level

Document Object Model

- Translated web pages often retain similar structure

```
<html>                        <html>
  <body>                        <body>
  <h1>
   Where is the cat?
  </h1>
   The cat sat on                El gato se sentó
   the mat.                      en la alfombra.
  </body>                       </body>
</html>                       </html>
```

- This includes links to the same images, etc.

Linearized Structure


```
[Start:html]
[Start:body]
[Start:h1]
[Chunk:17bytes]
[End:h1]
[Chunk:23bytes]
[End:body]
[End:html]
```

```
[Start:html]
[Start:body]
[Chunk:32bytes]
[End:body]
[End:html]
```

Levenshtein Alignment

[Start:html]	Keep
[Start:body]	Keep
[Start:h1]	Delete
[Chunk:17bytes]	Delete
[End:h1]	Delete
[Chunk:23bytes]	23 Bytes -> 32 Bytes
[End:body]	Keep
[End:html]	Keep

Content Similarity

- Simple things
  - same numbers or names in documents
  - often quite effective

- Use of lexicon
  - treat documents as bag of words
  - consider how many words in `EN` document have translations in `FR` document

- A bit more complex
  - semantic representations of documents content
  - bag of word vectors
  - neural network embeddings

- Major challenge: do this fast for $n \times m$ document pairs

Google's Content Matching

- Basic idea: translate everything into English, match large n-grams
- For each non-English document:
  1. Translate everything to English using MT
  2. Find distinctive ngrams
     (a) rare, but not too rare (5-grams)
     (b) used for matching only
- Build inverted index: ngram → documents

$$[\texttt{cat sat on}] \rightarrow \{\texttt{[doc}_1\texttt{, ES], [doc}_3\texttt{, DE], ...}\}$$
$$[\texttt{on the mat}] \rightarrow \{\texttt{[doc}_1\texttt{, ES], [doc}_2\texttt{, FR], ...}\}$$

Matching using Inverted Index

$$[\texttt{cat sat on}] \texttt{ -> } \{\texttt{[doc}_1\texttt{, ES], [doc}_3\texttt{, DE], ...}\}$$
$$[\texttt{on the mat}] \texttt{ -> } \{\texttt{[doc}_1\texttt{, ES], [doc}_2\texttt{, ES], ...}\}$$
$$[\texttt{on the table}] \texttt{ -> } \{\texttt{[doc}_3\texttt{, DE]}\}$$

- For each n-gram
  - generate all pairs where:
    * document list short (≤ 50)
    * source language different
- Result: `[doc`$_1$`, doc`$_3$`], ...`

Scoring using Forward Index

- Forward index maps documents to n-grams
- For each document pair `[`$\mathtt{d}_1$`, `$\mathtt{d}_2$`]`
  - collect scoring n-grams for both documents
  - build IDF-weighted vector
  - distance: cosine similarity

Scoring Document Pairs

- Given

$$\text{ngrams}(d_1) = n_1, n_2, ..., n_r$$
$$\text{ngrams}(d_2) = n'_1, n'_2, ..., n'_{r'}$$

- Inverse document frequency

$$\text{idf}(n) = \log \frac{|D|}{\text{df}(n)}$$

where: $|D|$ = number of documents
$\text{df}(n)$ = number of documents with n

- Scoring of IDF-weighted vectors v

$$v_{1,x} = \text{idf}(n_x) \text{ if } n_x \in \text{ngrams}(d_1), 0 \text{ otherwise}$$
$$v_{2,x} = \text{idf}(n_x) \text{ if } n_x \in \text{ngrams}(d_2), 0 \text{ otherwise}$$
$$\text{score}(d_1, d_2) = \frac{v_1 \dot{v}_2}{||v_1|| ||v_2|||}$$

sentence alignment

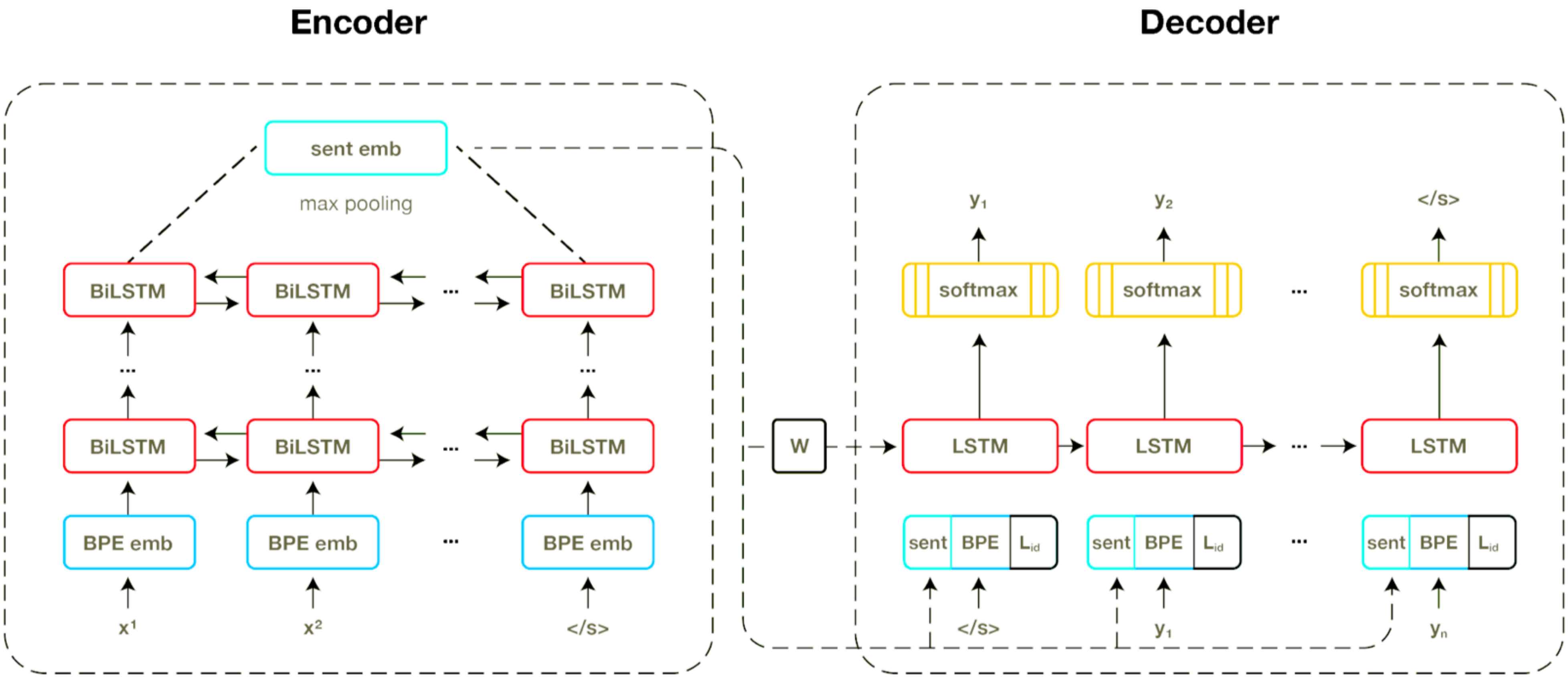
Sentence Alignment

- Much early work in 1990s, e.g., Gale and Church (1991)
  - find sequence of 1-1, 1-2, 0-1, etc., sentence alignment groups
  - good element in sequence = similar number of words
  - dynamic programming search for best sequence
- Featurized alignments
  - with dictionary (Hunalign)
  - with induced dictionary (Gargantua)
  - consider tags such as <P>
- Sensitive to noise — often large parts of page not translated

Sentence Pair Similarity

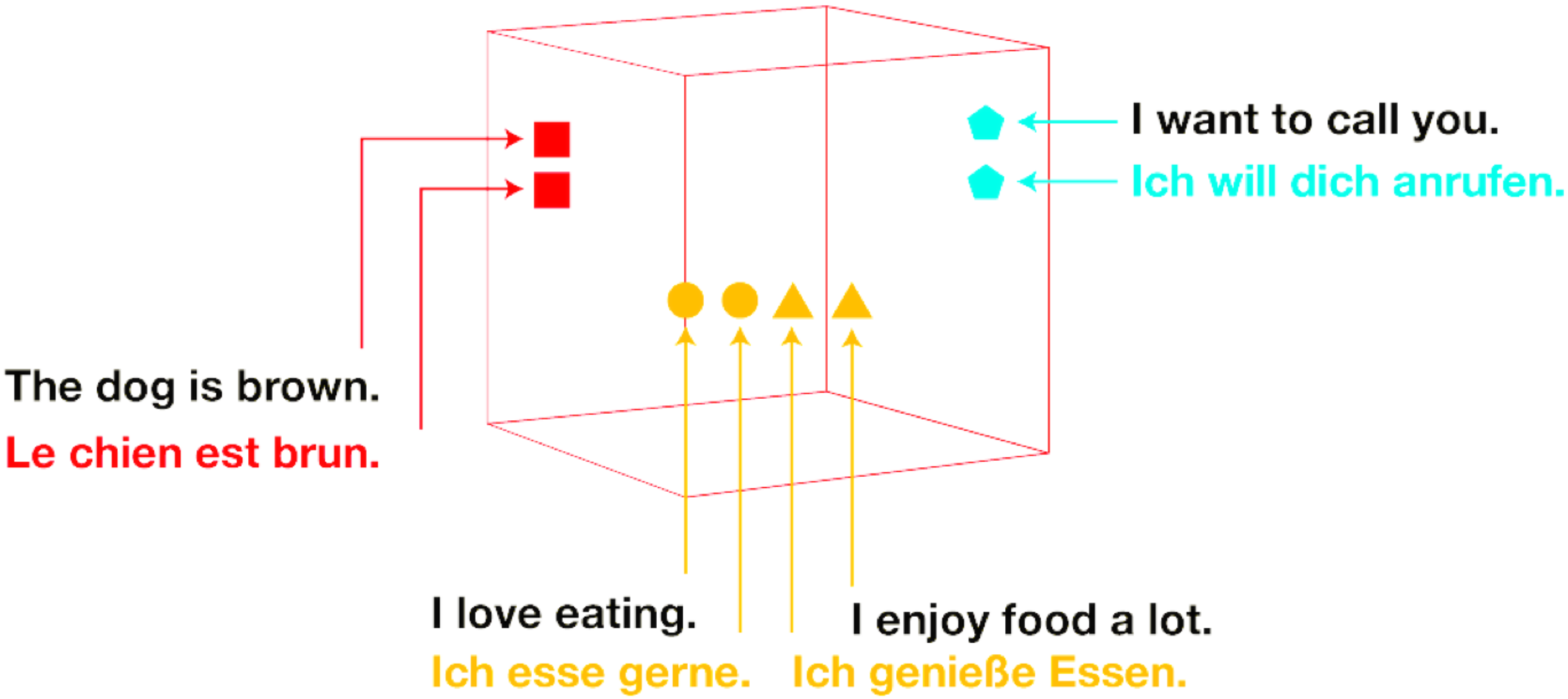
- Core Problem: both sentences must have same meaning
- Translate foreign sentence into English
  measure similarity with metrics like BLEU
- Words in one sentence have translation in the other
- Cross-lingual sentence embeddings

Sentence Embeddings

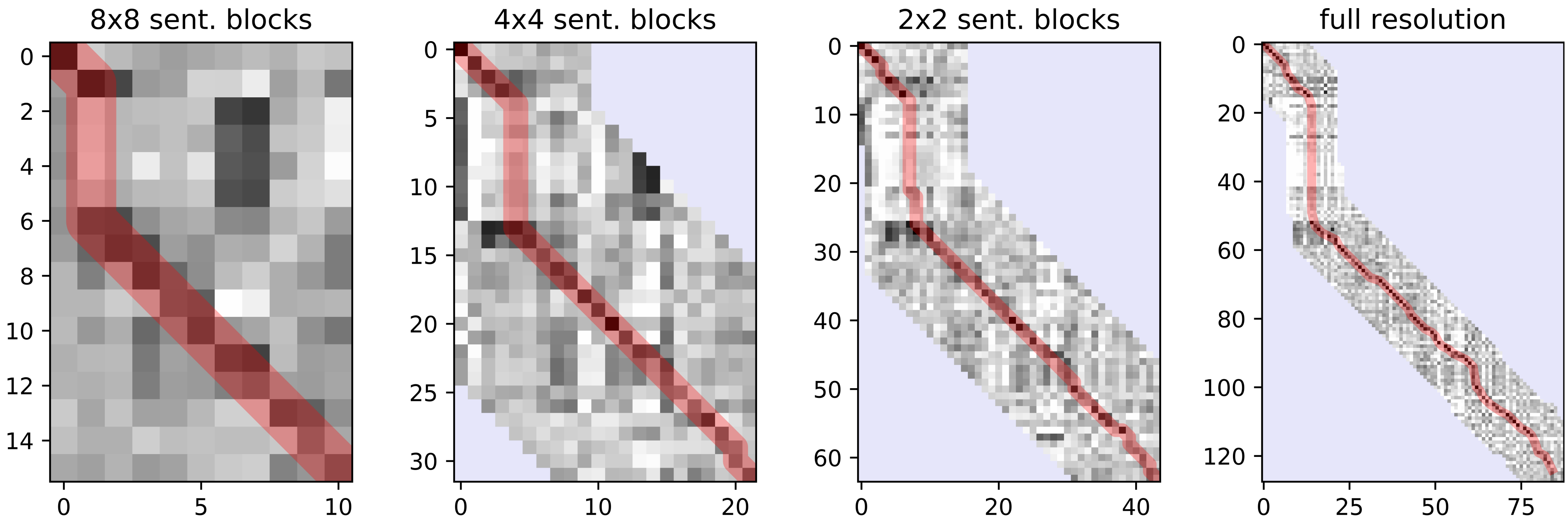


- LASER: Neural machine translation model with bottleneck feature

Sentence Embeddings



Vecalign



- Uses LASER sentence embeddings
- Linear time coarse-to-fine algorithm

sentence pair filtering

Filtering Bad Data

- Mismatched sentence pairs from errors in pipeline

- Non-literal translation
  e.g. news stories are notoriously non-literal

- Bad translations

- Machine translation
  - much of the parallel text on the Internet generated by Google Translate
  - detection hard — looks like very clean parallel data
  - maybe too clean (little reordering, very literal)
  - watermarking machine translation (Venugopal et al., 2011)

- How clean should it be?
  - trade-off between precision and recall unclear

Methods

- Dual cross-entropy
  - view sentence pair as input/output
  - score with neural machine translation model in both directions
  - scores should be low and similar
- LASER embeddings
- Feature-based approaches
  - matching numbers, named entities
  - language model probabilities
  - lexical translation probabilities
- Classifier
  - positive example: sentence pair from clean corpus
  - negative example: corrupted example (misalignment, words changed, ...)

Open Challenges

- Currently serious attempt at broad crawling for parallel data at JHU

- Major challenges
  - crawling (just using standard tool)
  - document alignment (major research topic)
    → shared task at WMT 2016 machine translation conference
  - sentence alignment (just using standard tool)
  - detection of machine translated text (some old work)
  - filtering out bad sentence pairs (major research topic)
    → shared tasks at WMT 2018–2020 machine translation conference

- JHU efforts (Paracrawl): continuously processing terabytes of data