Multilingual NLP

8. Word Alignment & Label Projection

Prof. Dr. Goran Glavaš
Center for AI and Data Science (CAIDAS), Uni Würzburg

Image: Alexander Mikhalchyk

After this lecture, you'll...

- Learn about machine-translation-based cross-lingual transfer
- Understand why it MT-based CL transfer is difficult for token-level tasks
- Learn about word alignment (WA) algorithms, symbolic and semantic
- Be aware of „mark-then-translate" as an alternative to WA

Content

- **Translation-Based CL Transfer**
- Word Alignment
  - Symbolic word alignment
  - Semantic word alignment
- Mark-then-Translate

Why Multilingual NLP?

- **Cross-Lingual transfer**: transfer supervised models for concrete NLP tasks
  - Models trained on labeled data in high-resource <u>source</u> language…
  - …make predictions on texts in low-resource <u>target</u> languages with little or no labeled data

Cross-Lingual Transfer: Practical Necessity

- Only a handful of NLP tasks have annotated data in many languages
  - Part-of-speech tagging (Universal Dependencies, UD)
  - Syntactic parsing (UD)
  - Named Entity Recognition (e.g., WikiANN)

- Higher-level semantic tasks often have only English training data
  - Generally more difficult tasks, e.g.:
    - Natural Language Inference (NLI)
    - Semantic Text Similarity (STS)
    - Question Answering (QA)
    - Causal Commonsense Reasoning
    - …

Translation-Based Transfer

- How about we use state-of-the-art **machine translation** to get annotated data in the languages we care about?

- Two common strategies:

1. **Translate train**
    - Automatically translate our training dataset in the source language L_S to the target language L_T
    - We obtain a (noisy) monolingual training dataset in L_T
    - We train a dedicated model M_T for L_T
    - For instances I_T from L_T we make predictions with M_T

Translation-Based Transfer

- How about we use state-of-the-art **machine translation** to get annotated data in the languages we care about?

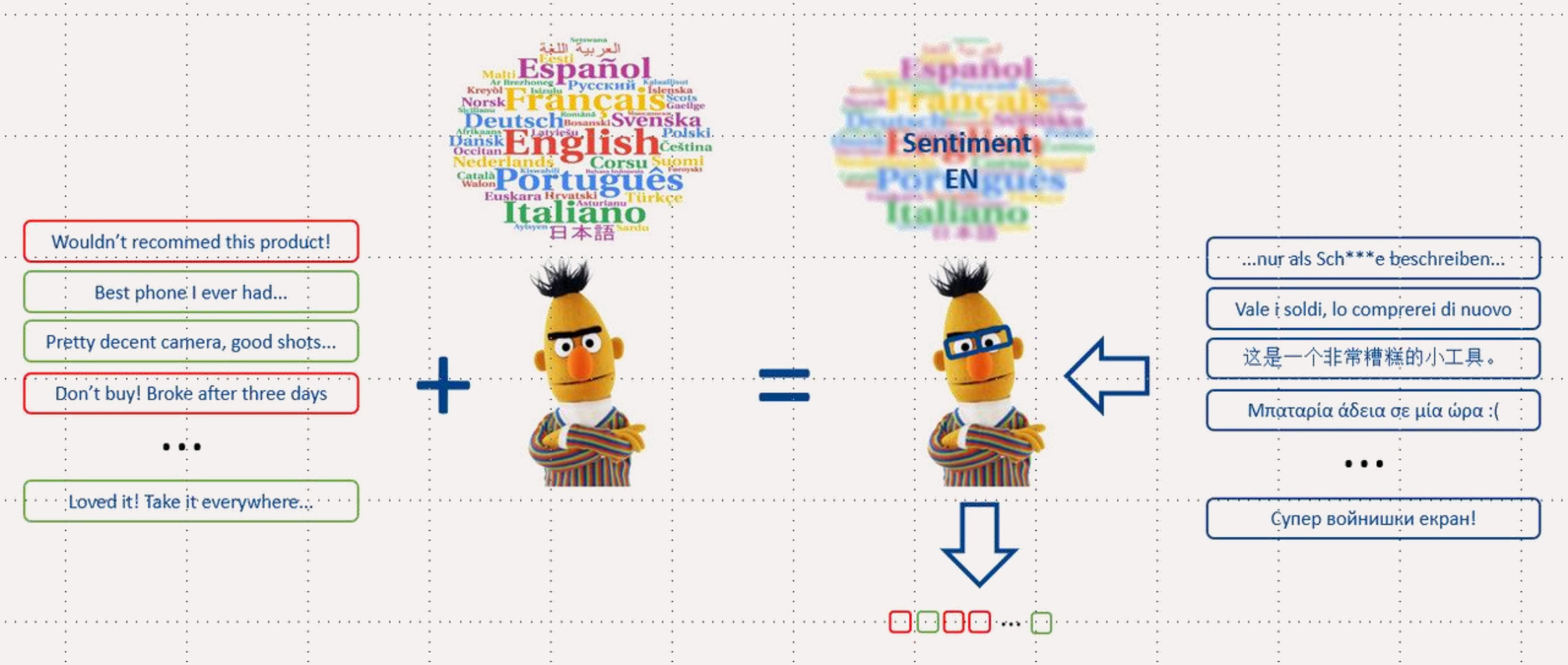
- Two common strategies:

2. Translate test

  - Train the model M_S using the clean training data in L_S
  - At inference time, for input I_T in L_T:
    - First translate I_T to the source language L_S
    - Make the prediction with the model M_S on the translation

Zero-Shot CL Transfer

- **Cross-lingual transfer with MMTs** is conceptually trivial
  1. Place a task-specific head on top of the Transformer body
  2. Perform standard fine-tuning using task-specific training data in L_S
  3. Use the Transformer and classifier to make predictions for data in L_T



Translate-Train Transfer

1. Automatically translate our training dataset in the source language L_S to the target language L_T

Training set (EN)		Noisy training set (HR)
Wouldn't recommed this product!		Ne bih preporučila ovaj proizvod...
Best phone I ever had...		Najbolji telefon koji sam ikad imao...
Pretty decent camera, good shots...	MT	Solidna camera, odlične slike...
Don't buy! Broke after three days		Ne kupujte! Pokvarilo se brzo...
...		...
Loved it! Take it everywhere...		Obožavam ga! Svugdje ga nosim...

Translate-Train Transfer

2. Train (i.e., fine-tune an MMT) on the translated train set
3. Make inference with the obtained model on target language input

Translate-Test Transfer

1. Train (i.e., fine-tune an MMT) on the original training set in L_S

Training set (EN)

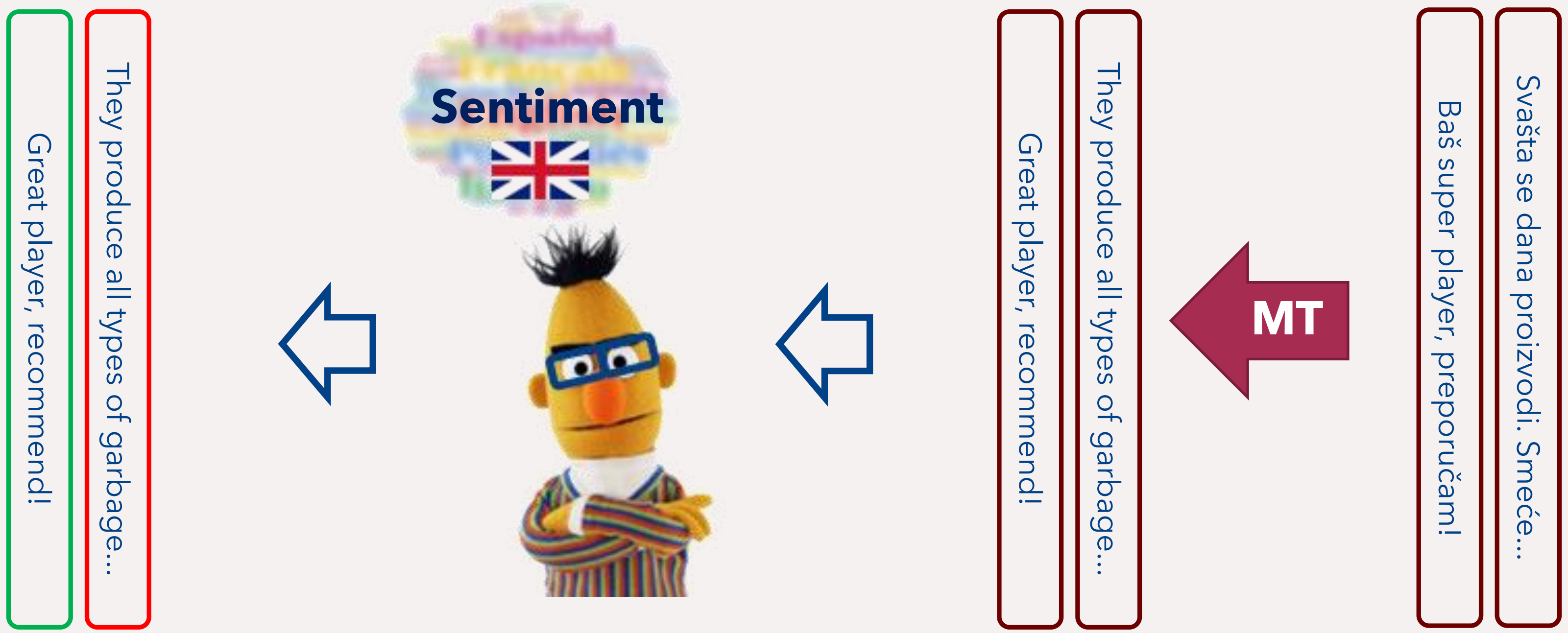
- Wouldn’t recommed this product!
- Best phone I ever had…
- Pretty decent camera, good shots…
- Don’t buy! Broke after three days

…

- Loved it! Take it everywhere…

Translate-Test Transfer

1. At inference, first translate the input from L_T to L_S
2. Then make prediction with the model trained on L_S data



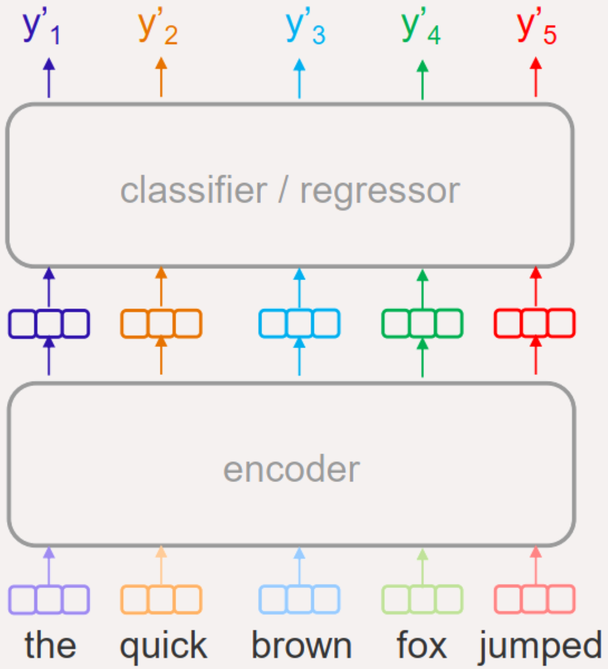
Translation-Based Transfer: Limitations

- Q: Translation-based vs. zero-shot CL transfer?
- Q: Translate-train vs. translate-test?
- Q: Shortcomings of translation-based transfer?
- The quality of translation-based transfer, obviously, depends on the quality of machine translation
- Translation-based CL transfer typically comparable or better than zero-shot CL transfer for higher-resource languages
  - For languages with strong MT models
- Translate-train typically more robust that translate test.
  - Especially for higher-level semantic tasks (QA, NLI, ...)
  - Q: Why?

Translation Transfer for Token Classification?

- **Token-level classification** (or regression), also known as sequence labeling, denotes tasks in which a label (class or score) is to be assigned to each input token
- Examples:
  - Part-of-speech tagging
  - Named entity recognition
  - Any of the other IE tasks where we need to extract
  - the span of tokens hat represent a concept instance
- Labels are at the token level
  - Translation-based transfer = need to align:
    - Words from the translation l_T
    - to the tokens of the source input l_S

    as tokens from l_S have labels



Translation Transfer for Token Classification?

- Example: Named Entity Recognition

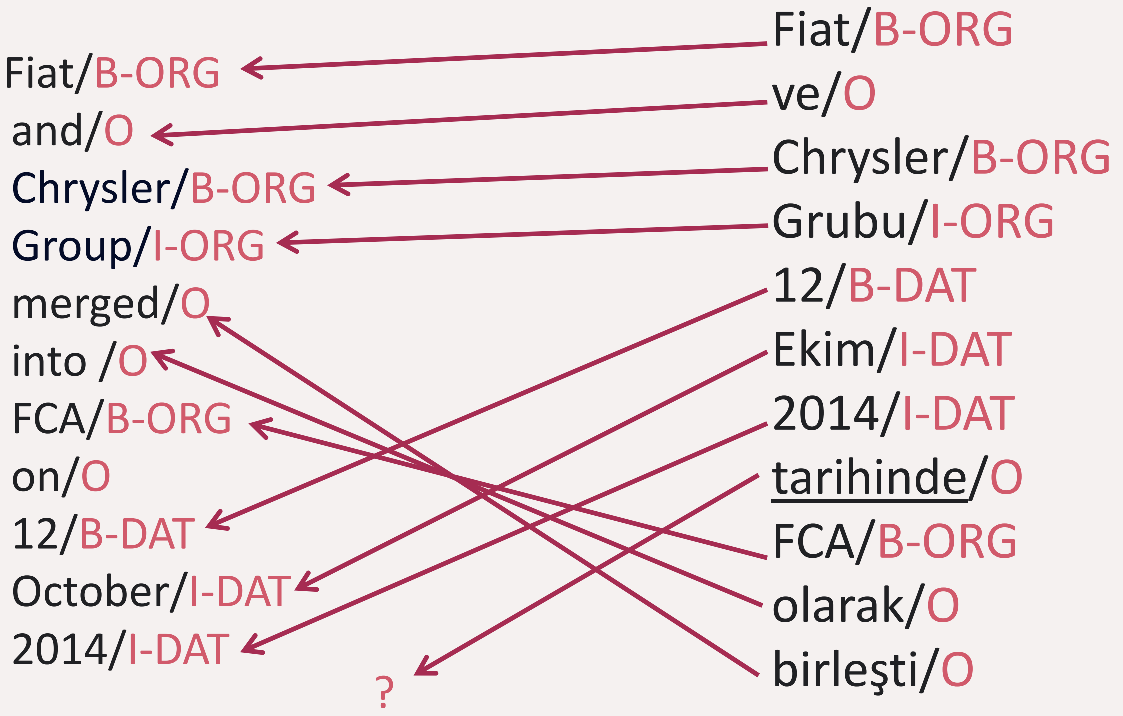
Fiat/B-ORG
and/O
Chrysler/B-ORG
Group/I-ORG
merged/O
into /O
FCA/B-ORG
on/O
12/B-DAT
October/I-DAT
2014/I-DAT

Machine translation EN->TR

Fiat/?
ve/?
Chrysler/?
Grubu/?
12/?
Ekim/?
2014/?
tarihinde/?
FCA/?
olarak/?
birleşti/?

Translation Transfer for Token Classification?

- To be able to transfer labels from l_S to its translation l_T (or vice-versa), we need to establish the **word alignment**
- This method of transfer is called annotation (or label) projection



Content

- Translation-Based CL Transfer
- **Word Alignment**
  - **Symbolic word alignment**
  - Semantic word alignment
- Mark-then-Translate

Word Alignment

- **Word alignment** is a task of finding mutual word translations between parallel texts (aka bitext), typically sentences that are translations of each other

- Word alignment is a „messy" task because:
  - 1-N, N-1, and N-N relations between words
  - Different word orders (and other morphosyntactic differences) between languages

- Word alignment was crucial in **statistical machine translation**
  - WA and SMT itself → two sides of the same coin
  - Shift to <u>NMT</u> reduced the importance of WA
    - Still important for CL transfer for token-level tasks!

Word Alignment

- **Word alignment** is practically (for CL transfer) defined as aligning each token t_i of the target sentence $\boldsymbol{t} = \{t_1, t_2, \ldots, t_m\}$ to a token s_j of the source sentence $\boldsymbol{s} = \{s_1, s_2, \ldots, s_n\}$

- In reality, not all target language tokens have a direct translation in the source sentence → we introduce a special „empty" token s_0
  - $\boldsymbol{s} = \{\underline{s_0}, s_1, s_2, \ldots, s_n\}$

- Multiple tokens from $\boldsymbol{t}$ can be aligned to the same source token s_j

Word Alignment

- **IBM Word Alignment Models**
  - Originally SMT models
  - Later on primarily used as word alignment models

Translation formulation:

- We're going to swap source and target language for a moment
  - Source in the translation formulation is target for alignment
- We're searching for the most likely translation ***s*** for a given input ***t***

$$s^* = \operatorname{argmax}_{\mathbf{s}} P(\mathbf{s}|\mathbf{t}) \quad \text{(which, given Bayes rule)}$$
$$\propto \operatorname{argmax}_{\mathbf{s}} \underbrace{P(\mathbf{t}|\mathbf{s})}_{\text{Translation model}} \; \underbrace{P(\mathbf{s})}_{\text{Language model (of translation target language, } L_S)}$$

Word Alignment

- **IBM Models**
  - Translation model: estimates the probabilities P(**t**|**s**)
  - IBM Model 2:
    - Let's assume alignments $a_1, a_2, \ldots, a_m$
    - Alignment $a_i = (i, j)$ – means that t_i is aligned to s_j

$$\mathrm{P}(\mathbf{t}|\mathbf{s}) = \prod_{i=1}^{m} q_p(\underbrace{j|i}_{a_i}, m, n) * q_w(t_i|s_j)$$

Position alignment score (for positions i and j given lengths m and n)

Word translation scores (regardless of positions of words)

Word Alignment: IBM Model 2

- If we had estimates $q_p(j|i,m,n)$ for all position pairs i and j
- And estimates $q_w(t|s)$ for all word pairs

- We could then easily compute the „optimal" word alignment (according to the IBM Model 2) for any two parallel sentences **t** and **s**

- **Algorithm**
  - For each t_i in ***t***
    - Select s_j in ***s*** for which $q_p(j|i,m,n) * q_w(t_i|s_j)$ is the largest

Word Alignment: IBM Model 2

- Q: How do we obtain position alignment scores q_p and word translation scores q_w?
- We estimate them from the **parallel corpus** using an expectation maximization (EM) algorithm
- Parallel corpus: $\{\mathbf{s}^{(k)}, \mathbf{t}^{(k)}\}_k$

Word Alignment: IBM Model 2

- Q: How do we obtain position alignment scores q_p and word translation scores q_w?
- Parallel corpus: $\{\mathbf{s}^{(k)}, \mathbf{t}^{(k)}\}_k$
- Let's for a moment assume that we also have „gold" word alignments in our training corpus (which in reality, we won't have)
  - We can directly do the maximum likelihood estimation (MLE) of q_w and q_p as follows (function „c" indicates the raw count):

$$q_p(j|i,m,n) = \frac{c(j|i,m,n)}{c(i,m,n)}$$

Number of times (in our training parallel corpus with gold alignments) that the i-th word in $\mathbf{t}$ (which is of length m) was aligned with the j-th word in $\mathbf{s}$ (which is of length n)

Word Alignment: IBM Model 2

- Q: How do we obtain position alignment scores q_p and word translation scores q_w?
- Parallel corpus: $\{\mathbf{s}^{(k)}, \mathbf{t}^{(k)}\}_k$
- Let's for a moment assume that we also have „gold" word alignments in our training corpus (which in reality, we won't have)
  - We can directly do the maximum likelihood estimation (MLE) of q_w and q_p as follows (function „c" indicates the raw count):

$$q_w(t|s) = \frac{c(t,s)}{c(s)}$$

Number of times some target word (e.g., *Hund*) was aligned to some source word (e.g., *dog*)

Number of times that source word (e.g., *dog*) appeared in the parallel corpus

Word Alignment: IBM Model 2

- Q: How do we obtain position alignment scores q_p and word translation scores q_w?
- In reality, we won't have word alignments provided on our parallel corpus $\{\mathbf{s}^{(k)}, \mathbf{t}^{(k)}\}_k$
- We cannot really count $c(t|s)$ and $c(j|i,m,n)$
- But words and positions of alignments will tend to appear more often over the sentences of our parallel corpus
- „Learning" algorithm: a variant of **expectation maximization,** iteratively:
  1. Estimate changes to counts $c(t|s)$ and $c(j|i,m,n)$ (expected counts) from current parameter values (q_p and q_w)
  2. Update all parameters (q_p and q_w) based on new expected counts

Word Alignment: IBM Model 2

- Parallel corpus $\{\mathbf{s}^{(k)}, \mathbf{t}^{(k)}\}_k$

- Let's assume some parameter initialization $q_p(j|i,m,n)$, $q_w(t|s)$, e.g., with random values

- The EM algorithm then iterates over each sentence pair $\mathbf{s}^{(k)}, \mathbf{t}^{(k)}$ and:
  - Computes the probability of alignment $\delta(k, i, j)$ for positions i (from $\mathbf{t}^{(k)}$) and j (from $\mathbf{s}^{(k)}$) as follows:

$$\delta(k, i, j) = \frac{q_p(j|i, m^k, n^k) * q_w(t_i{}^k | s_j{}^k)}{\sum_{j'=0}^{n} q_p\left(j' \middle| i, m^k, n^k\right) * q_w(t_i{}^k | s_{j'}{}^k)}$$

Word Alignment: IBM Model 2

$$\delta(k, i, j) = \frac{q_p(j|i, m^k, n^k) * q_w(t_i^{\,k}|s_j^{\,k})}{\sum_{j'=0}^{n} q_p(j'|i, m^k, n^k) * q_w(t_i^{\,k}|s_{j'}^{\,k})}$$

<u>Alignment algorithm</u>: (initialize all parameters (q_p, q_w), e.g., to random values)

- For step in 1 to S (S iterations of the algorithm):
  - Initialize all counts ($c(j|i,m,n)$ and $c(i,m,n)$, $c(t_i^{\,k}, s_j^{\,k})$, $c(s_j^{\,k})$) to **zero**
  - **for each** training pair of sentences $\mathbf{t}^{(k)}$ and $\mathbf{s}^{(k)}$:
    - **for** i in 1 to m^k (iterating over all tokens of $\mathbf{t}^{(k)}$):
      - **for** j in 0 to n^k (iterating over all tokens of $\mathbf{s}^{(k)}$):
        - Compute δ(k, i, j) according to the above formula
        - Update count expectations:
          - $c(j|i, m^k, n^k) \leftarrow c(j|i, m^k, n^k) + \delta(k, i, j)$
          - $c(i, m^k, n^k) \leftarrow c(i, m^k, n^k) + \delta(k, i, j)$
          - $c(t_i^{\,k}, s_j^{\,k}) \leftarrow c(t_i^{\,k}, s_j^{\,k}) + \delta(k, i, j)$
          - $c(s_j^{\,k}) \leftarrow c(s_j^{\,k}) + \delta(k, i, j)$
  - Update the parameters based on collected (expected) counts
    - $q_p(j|i, m, n) = \frac{c(j|i,m,n)}{c(i,m,n)}$ and $q_w(t|s) = \frac{c(t,s)}{c(s)}$

Word Alignment: IBM Model 2

- Q: Why does this (intuitively) work?
  - Words that are translations of each other will appear in multiple pairs of sentence translations
  - Thus their count accumulation $c(t, s)$ will be larger

- Based on morpho-syntactic similarities/differences between languages a „more informed" initialization of the positional alignments q_p possible
  - E.g., if the languages have same word order → $q_p(j|i, m, n)$ can be set larger for values of i and j that are closer to each other

Word Alignment: FastAlign

Dyer, C., Chahuneau, V., & Smith, N. A. (2013, June). A simple, fast, and effective reparameterization of IBM model 2. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 644-648).

- IBM Model 2 is „sparse" and has very many parameters
  - $q_p(j|i,m,n)$ → i*j parameters for every different combination of lengths of sentences in the training set (every different m-n combination)
  - Likelihood of aligning certain position i and j is probably similar for various sentence lengths m and n
- **FastAlign** is a sparse WA model that reduces the number of parameters
  - Essentially a „reparametrization" of IBM Model 2

Word Alignment: FastAlign

Dyer, C., Chahuneau, V., & Smith, N. A. (2013, June). A simple, fast, and effective reparameterization of IBM model 2. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 644-648).

- **FastAlign** is a sparse WA model that reduces the number of parameters
  - Essentially a „reparametrization" of IBM Model 2
  - Instead of having an (updatable) parameters $q_p(j|i,m,n)$ for each combination (i, j, m, n) combination, we compute it with a function:

$q_p(j|i,m,n)$ = p_0 if $j = 0$ (i.e., p_0 is the probability of no alignment)

$(1 - p_0) * \frac{\exp(\lambda * h(i,j,m,n))}{\sum_{j'=1}^{n} \exp(\lambda * h(i,j,m,n))}$ otherwise ($j > 0$)

Word Alignment: FastAlign

Dyer, C., Chahuneau, V., & Smith, N. A. (2013, June). A simple, fast, and effective reparameterization of IBM model 2. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 644-648).

- **FastAlign** is a sparse WA model that reduces the number of parameters
  - Essentially a „reparametrization" of IBM Model 2
    - $q_p(j|i,m,n) = p_0$ if $j = 0$ (i.e., p_0 is the probability of no alignment)

$$(1 - p_0) * \frac{\exp(\lambda * h(i,j,m,n))}{\sum_{j'=1}^{n} \exp(\lambda * h(i,j,m,n))} \text{ otherwise } (j > 0)$$

  - Where h is a fixed function of relative positional distance:
    - $h(i, j, m, n) = -|i/m - j/n|$
    - Larger h → lower probability of alignment between positions i (in ***t***) and j (in ***s***)

Word Alignment: FastAlign

Dyer, C., Chahuneau, V., & Smith, N. A. (2013, June). A simple, fast, and effective reparameterization of IBM model 2. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 644-648).

- **FastAlign** is a sparse WA model that reduces the number of parameters
  - Essentially a „reparametrization" of IBM Model 2
    - $q_p(j|i,m,n) = p_0$ if $j = 0$ (i.e., p_0 is the probability of no alignment)

$$(1 - p_0) * \frac{\exp(\lambda * h(i,j,m,n))}{\sum_{j'=1}^{n} \exp(\lambda * h(i,j,m,n))} \text{ otherwise } (j > 0)$$

  - λ (≥ 0): decides how strongly we prefer alignments of close positions
    - $\lambda < 1$: scales down the effect of relative distance of i and j
      - Appropriate for syntactically dissimilar languages
    - $\lambda > 1$: emphasizes the effect of relative distance of i and j
      - Appropriate for syntactically similar languages

Word Alignment

- Problems with „symbolic" word alignment methods
  - Same as for any other NLP task/problem

  - Do not capture semantic relations between words
    - Probability/count of alignment q_w(car, Auto) is independent of the probability of alignment q_w(automobile, Auto)

  - Strictly requires parallel data
    - The more the better
    - Hard to find/create large parallel corpora for low-resource langs
    - Not able to align words that are not in the parallel „training" corpus

Content

- Translation-Based CL Transfer
- **Word Alignment**
  - Symbolic word alignment
  - **Semantic word alignment**
- Mark-then-Translate

Semantic Word Alignment

- We assume we have a semantic representation for each t_i from ***t*** and each word s_j from ***s***
- The representations of words from the target language L_T need to be semantically aligned with the representations of words from the source language L_S
- Q: how can we obtain embeddings that satisfy this?
  1. Cross-lingual word embedding spaces (CLWEs)
  2. Multilingual Transformers (e.g., mBERT)

Semantic Word Alignment

- Let $\boldsymbol{t}_i \in \mathbb{R}^d$ be an embedding of the token t_i from $\boldsymbol{t}$
- Let $\boldsymbol{s}_j \in \mathbb{R}^d$ be an embedding of the token s_j from $\boldsymbol{s}$

- Then we can obtain the similarity matrix $S \in \mathbb{R}^{m \times n}$ which contains cosine similarities between all vectors $\mathbf{t}_i$ from $\boldsymbol{t}$ and $\mathbf{s}_j$ from $\boldsymbol{s}$
  - $S_{ij} = \cos(\mathbf{t}_i, \mathbf{s}_j)$
- We use the similarity matrix S to obtain the alignments:
  - Greedy alignment
  - Greedy alignment with removal
  - Optimal alignment (with removal)
- Does <u>not require</u> parallel data

Semantic Word Alignment

- **Greedy alignment**
  - For each word t_i we find the s_j that is semantically most similar to t_i according to cosine similarity between their embeddings: $\cos(\mathbf{t}_i, \mathbf{s}_j)$
  - I.e., in each row $S[i:]$, we find the cell S_{ij} with the maximal value
  - The same column j (i.e., same word s_j) may be chosen for multiple rows (i.e., multiple words t_i may be aligned to the same s_j)

Semantic Word Alignment

- **Greedy alignment with removal**
  - Iteratively:

1. Find the most similar pair (**t_i**, **s_j**), i.e., the cell S_{ij} with the maximal value (among the remaining eligible cells) and make the alignment (t_i, s_j)

2. Prevent any further alignments that involve either t_i or s_j
   - I.e., set all values in S[i, :] and S[:, j] to -1 (minimal cosine)

Semantic Word Alignment

- **Optimal alignment** (with removal)
  - We are solving the following optimization problem: we're looking for a set of alignment that maximizes the sum of pairwise similarities
  - Let A be a binary matrix (values 0 or 1): $A_{ij} = 1$ indicates that an alignment has been established between t_i and s_j
    - Constraint: A can have only one „1" in each row and each column

$$A^* = \mathrm{argmax}_{A \in \{0, 1\}^{m \times n}} \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} S_{ij}$$

Semantic Word Alignment

- **Optimal alignment** (with removal)

$$A^* = \text{argmax}_{A \in \{0,1\}^{m \times n}} \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} S_{ij}$$

- This is a well-known problem called bipartite graph matching
  - Also known as (optimal) alignment problem

- Efficient algorithms exist (polynomial time)
  - The Hungarian algorithm (Kuhn-Munkres algorithm, from 1955) – solved the problem in $O(n^4)$
  - Later – better algorithms with complexity $O(n^3)$

Content

- Translation-Based CL Transfer
- Word Alignment
  - Symbolic word alignment
  - Semantic word alignment
- **Mark-then-Translate**

Mark-then-Translate

- **Simple idea:** put special tags around the tokens, indicating their label and translate with some MT system
  - Used a lot in token-level CL transfer, but thouroughly empirically tested only recently

<ORG>Fiat</ORG> and <ORG>Chrysler Group</ORG> merged into <ORG>FCA</ORG> on <DAT>12 October 2014</DAT>

MT
EN→TR

(ideally)

<ORG>Fiat</ORG> ve <ORG>Chrysler Grubu </ORG> <DAT>12 Ekim 2014 </DAT> tarihinde <ORG>FCA</ORG> olarak birleşti

Mark-then-Translate

Chen, Y., Jiang, C., Ritter, A., & Xu, W. (2022). Frustratingly Easy Label Projection for Cross-lingual Transfer. arXiv preprint arXiv:2211.15613.

- More directly dependent on the quality of MT (i.e., abundance of parallel data between languages)
- MtT gets better as MT models get better
  - Chen et al. (2023) experiment with Google Translate and open-source NLLB („No Language Left Behind", covered in L9)
  - Report MtT better than WA-based label projection for many languages and tasks
    - Though mostly for high- and moderate-resource languages

Image: Alexander Mikhalchyk