11-731 (Spring2013)
Lecture 22:

Example-Based Machine Translation

Ralf Brown
9 April 2013

What is EBMT?

- A family of data-driven (corpus-based) approaches
  - Can be purely lexical or involve substantial analysis
- Many different names have been used
  - Memory-based, case-based, experience-guided
- One definining characteristic:
  - Individual training instances are available at translation-time

Early History of EBMT

- First proposed by Nagao in 1981
  - “translation by analogy”
  - Matched parse trees with each other
- DLT system (Utrecht)
  - “Linguistic Knowledge Bank” of example phrases
- Many early systems were intended as a component of a rule-based MT system

A Sampling of EBMT Systems

- ATR (Sumita) 1990,1991
- CTM, MBT3 (Sato) 1992, 1993
- METLA-1 (Juola) 1994, 1997
- Panlite / CMU-EBMT (CMU: Brown) 1995-2011
- ReVerb (Trinity Dublin) 1996-1998
- Gaijin (Dublin City University: Veale & Way) 1997
- TTL (Öz, Güvenir, Cicekli) 1998
- Cunei (CMU: Phillips) 2007-

EBMT and Translation Memory

- Closely related, but different focus
  - TM is an interactive tool for a human translator, while EBMT is fully automatic
- TM systems have become more EBMT-like
  - Originally simply presented the best-matching complete sentence to the user for editing
  - Can now retrieve and re-assemble fragments from multiple stored instances

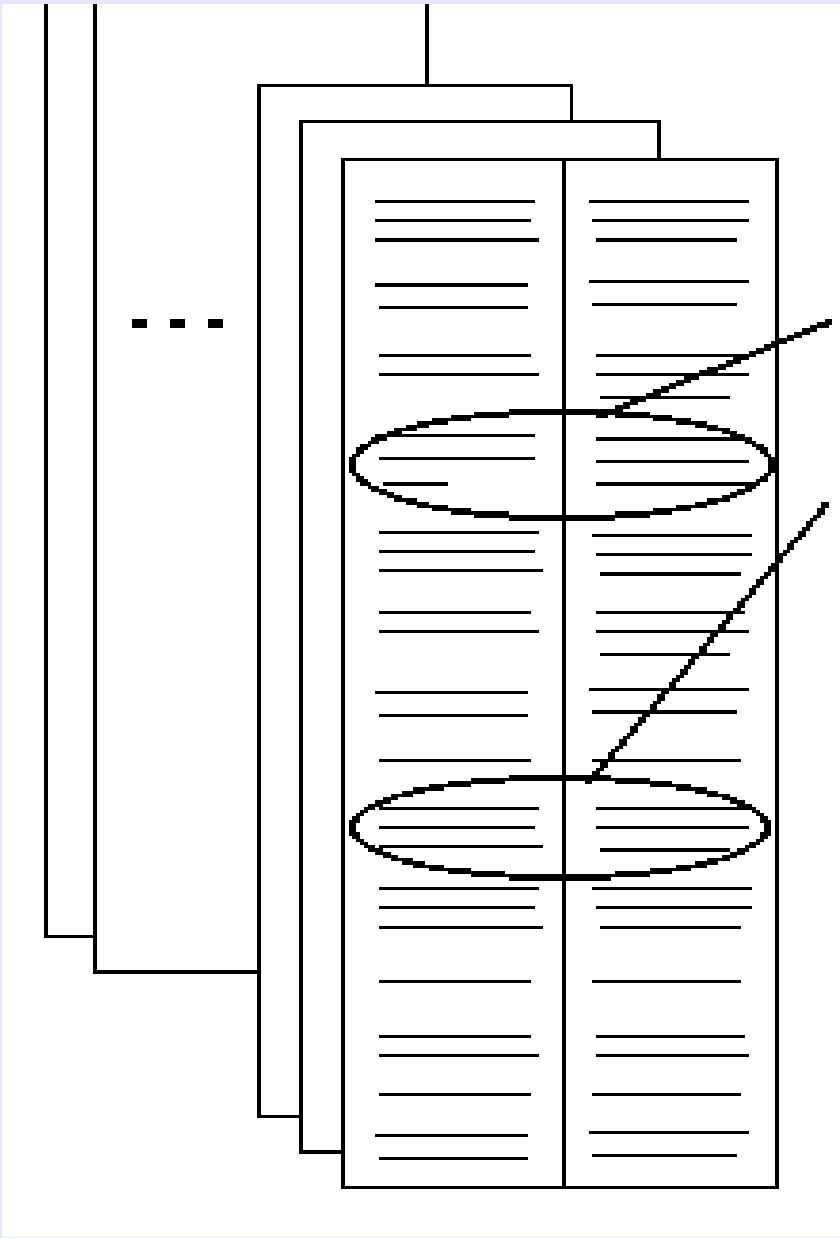
EBMT and Phrase-Based SMT

- PBSMT is very similar to lexical EBMT using arbitrary-fragment matching
  - This style of EBMT can be thought of as generating an input-specific phrase table on the fly
- EBMT can guarantee that input identical to a training example generates the identical translation in the corpus
  - PBSMT only if the input is less than the maximum phrase length in the phrase table

EBMT Workflow

- Three stages
  - Segment the input
  - Translate and adapt the input segments
  - Recombine the output
- One or more stages may be trivial in a given system

Sample Translation Flow



New Sentence (Source)

Yesterday, 200 delegates met with President Obama.

Matches to Source Found

Yesterday, 200 delegates met behind closed doors…	Gestern trafen sich 200 Abgeordnete hinter verschlossenen Türen…
Difficulties **with President Obama**…	Schwierigkeiten mit Praesident Obama…

Alignment (Sub-sentential)

Yesterday, 200 delegates met behind closed doors…	**Gestern trafen sich 200 Abgeordnete** hinter verschlossenen…
Difficulties with President Obama over…	Schwierigkeiten **mit Präsident Obama**…

Translated Sentence (Target)

Gestern trafen sich 200 Abgeordnete mit Präsident Obama.

Segmenting the Input

- No segmentation: retrieve best-matching complete training instance
- Linguistically-motivated
  - Parse-tree fragments
  - Chunks / Marker Hypothesis
- Arbitrary word sequences
  - like PBSMT

Marker Hypothesis

- (Green, 1979) proposed psycholinguistic universal
  - All languages are marked for grammar by a closed set of specific lexemes and morphemes
- Used by multiple MT systems from Dublin City University
  - Multiple classes such as PREP, DET, QUANT
  - Members of marker class signal begin/end of phrase
  - Phrases are merged if the earlier one is devoid of non-marker words

Parse-Tree Fragments

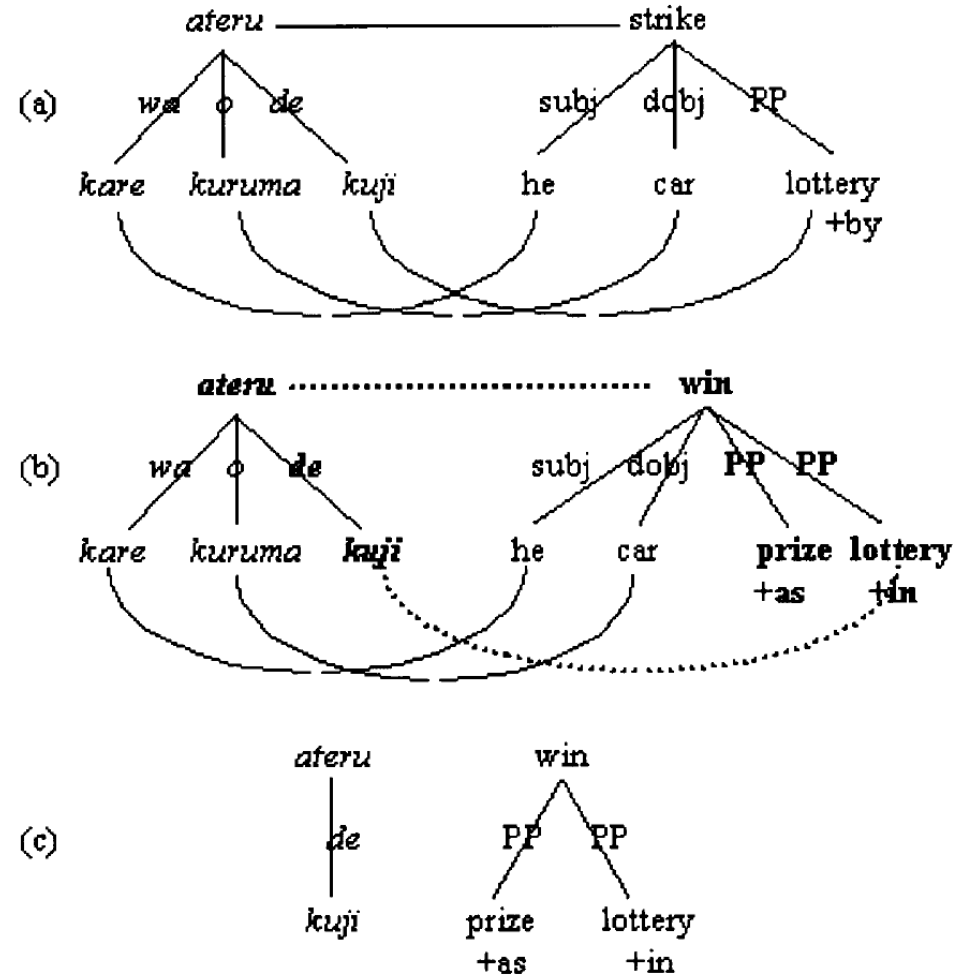


Figure 6. Extraction of translation pattern from example. (Watanabe 1993: 293).

Translating Input Fragments

- Determine the target text corresponding to the matched portion of the example
  - Word-level alignment techniques for strings
  - Node-matching techniques for parse trees
- Apply any fix-ups needed as a result of fuzzy matching
  - Word replacement or morphological inflection
  - Filling gaps using other fragments

Recombining the Output

- None, if full example matched
- Simple concatenation
- Dynamic-programming lattice search
- SMT-style stack decoder with language models

Finding Matching Examples

- Important for scalability to have fast lookups
- Most EBMT systems apply database techniques
  - Early systems used inverted files, relational databases
  - Suffix arrays now in common use

Suffix Arrays

- Corpus is treated as one long string and sorted lexically starting at every word
- O(k log n) lookups for k-grams
  - All instances of a k-gram are represented by a simple range in the index
  - Can find all matches of any length in a single pass
- Can be transformed into a self-index which does not require the original text
  - Indexed corpus can be smaller than the original text

Suffix Array Example (1)

Indexing “Albuquerque” by characters:

```
 0   A l b u q u e r q u e $
 1   l b u q u e r q u e $ A
 2   b u q u e r q u e $ A l
 3   u q u e r q u e $ A l b
 4   q u e r q u e $ A l b u
 5   u e r q u e $ A l b u q
 6   e r q u e $ A l b u q u
 7   r q u e $ A l b u q u e
 8   q u e $ A l b u q u e r
 9   u e $ A l b u q u e r q
10   e $ A l b u q u e r q u
11   $ A l b u q u e r q u e
```

Suffix Array Example (2)

Sort lexically, remembering original location:

```
 0   A l b u q u e r q u e $
 2   b u q u e r q u e $ A l
 6   e r q u e $ A l b u q u
10   e $ A l b u q u e r q u
 1   l b u q u e r q u e $ A
 4   q u e r q u e $ A l b u
 8   q u e $ A l b u q u e r
 7   r q u e $ A l b u q u e
 5   u e r q u e $ A l b u q
 9   u e $ A l b u q u e r q
 3   u q u e r q u e $ A l b
11   $ A l b u q u e r q u e
```

Suffix Array Example (3)

Array of original positions is our index; use it to indirect into the original text

```
0 2 6 10 1 4 8 7 5 9 3 11

A l b u q u e r q u e $
```

Lookups are binary searches via the indirection of the index

Burrows-Wheeler Transform

- Convert suffix array into a self-index by generating a vector of successor pointers
- After storing an index of the starting position of each type in the corpus, we can throw away the original text
- BWT index can be stored in a compressed form for even greater space savings

Burrows-Wheeler Transform

0	4	A
1	10	b
2	7	e
3	11	e
4	1	l
5	8	q
6	9	q
7	6	r
8	2	u
9	3	u
10	5	u
11	0	$

- Match for single char is its range
- Extend match to the left by finding the range of entries whose successors lie within the range of the match
  - 'e' is rows 2-3
  - for 'ue', 'u' is rows 8-10, of which 8 and 9 point within 2-3
- Each extension takes two binary searches because successors are sorted

Suffix Array Drawbacks

- Some additional housekeeping overhead to retrieve complete training example
- Fuzzy / gapped matching is slow
  - Can degenerate to O(kn)
- Incremental updates are expensive
  - Workaround is to have a second, small index for updates

Fuzzy Matching

- Increase the number of candidates by permitting substitution of words
  - source-language synonym sets
  - common words for rare words (e.g. “bird” for “raven”)
- In the limit, leave a gap and allow any word
  - like Hiero

Generalizing Examples

- Can also generalize the example base and match using generalizations
- Equivalence classes
  - index “Monday”, “Tuesday”, etc. as <weekday>
  - look up <weekday> for “Monday” etc. in the input
- Base forms
  - index “is”, “are”, etc. as “be” plus morphological features
  - match on base forms and use morphology to determine best matches

System: ReVerb (1)

- Views EBMT as Case-Based Reasoning
- Addresses translation divergences by establishing links based on lexical meaning, not part of speech
  - but POS mismatches are penalized
- Corpus is tagged for morphology, POS, and syntactic function, then manually disambiguated
- “Adaptability” scores penalize within- or cross-language dependencies

System: ReVerb (2)

- Adaptability levels:
  - 3: one-to-one SL:TL mapping for all words
  - 2: syntactic functions map, but not all POS tags
  - 1: different functions, but lexical equivalence holds
  - 0: unable to establish correspondence
- Generalization (“Case templatization”)
  - Substitute POS tags for chunks that are mappable at a given adaptability level

System: ReVerb (3)

- Retrieval in two phases
  - Exact lexical matches
  - Add head matches and instances with good mappability
- Run-time adaptation of TL based on dependency, not linear order
  - Divergent fragment is replaced using correponding TL from case base
  - Errors can be user corrected and stored as new cases

System: Gaijin

- German-English translation
  - Corpus of 1836 sentence pairs in software domain
- Uses Marker Hypothesis to segment input
- Sentences converted to marker sequences for matching
- Adaptation via “grafting” (replacing entire marker-based phrase) and “keyhole surgery” (replacing an individual word)

System: CMU-EBMT

- Lexical system with optional generalization
  - recursive generalization supported, yielding equivalent of SCFG
- Arbitrary-sequence matching
- Contextual features
- Generalized matches are adapted by inserting appropriate translation for generalization
- SMT-style decoder for recombination
  - Can perform additional adaptation by filling in TL gaps through overlapping segments

CMU-EBMT Origins

- The earliest implementation (1992) was in Lisp as part of the Pangloss system; various approaches to matching were tried
- A second implementation in C was begun to replace the index-lookup code from the Lisp version for greater speed
  - this version already performed contiguous-phrase matching, but conflated all function words in matching
- The current C++ implementation was begun in 1995 as a complete replacement for the Lisp/C hybrid

Pangloss-Lite (1995)

- The Pangloss system was a full suite called the Translator's Workstation, including post-editing facilities, visualization, etc.
- TWS implemented in Lisp, loaded Prolog-based Panglyzer KBMT system, C-based EBMT/LM
- very slow start-up times (15 minutes!), so implemented a simple wrapper around EBMT, LM, and a dictionary
  - resulted in a translation system that started up in a few seconds

TIDES (2002-2005)
GALE (2005-2008)

- Focus shifts towards huge training corpora
  - 200+ million words
  - (TIDES originally had a 100k-word small data track, quickly dropped)
- Goal is maximum translation quality with little or no regard for resource requirements (including translation time – minutes per sentence is OK)

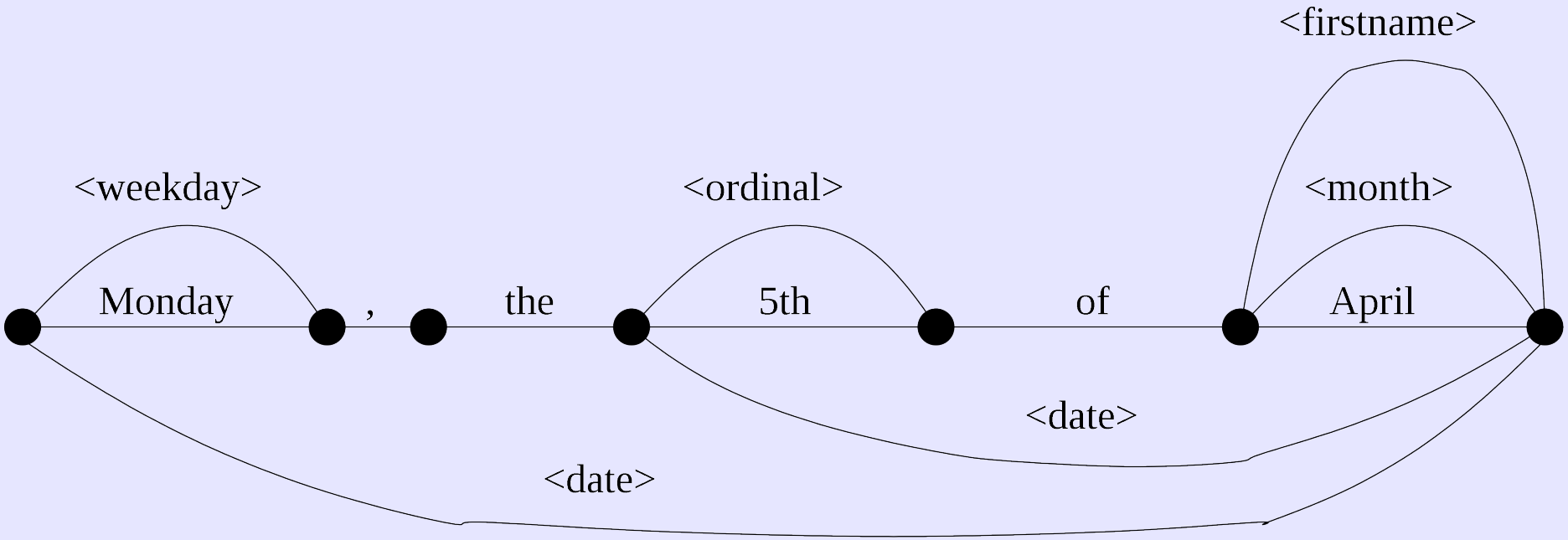
EBMT Adapts to TIDES/GALE

- Can take a lattice of hypotheses as input
- New index: Burrows-Wheeler transformed version of a suffix array for better scalability
- Many more matches are processed, and translation score is now a combination of alignment score, translation probability, and contextual weighting
- Training examples can be generalized, but that capability sees little use on large corpora

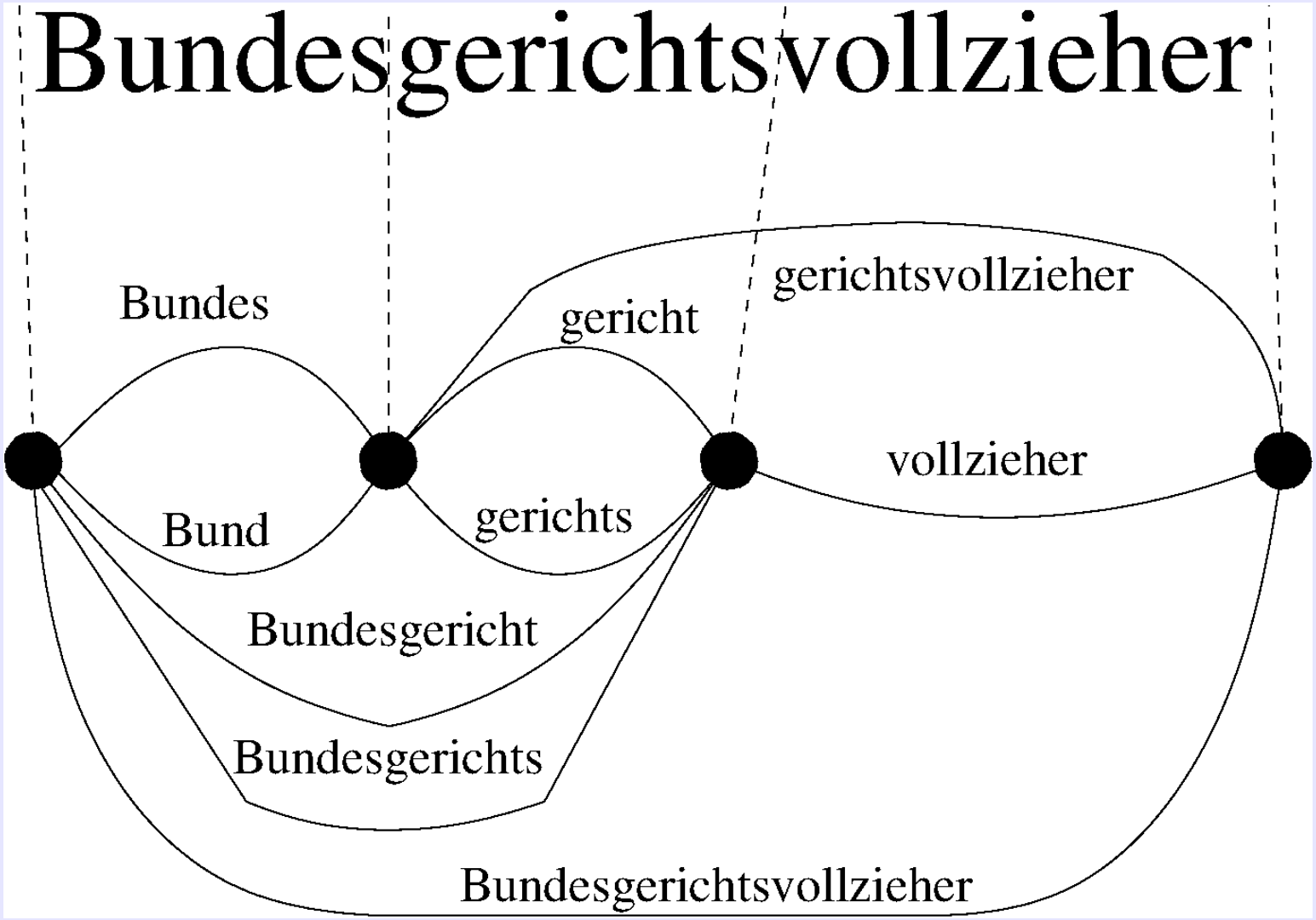
Lattice Processing

- Sometimes, the input is ambiguous, and we want to preserve that ambiguity
  - multiple word segmentations for Asian languages
  - confusion networks or word lattices from speech recognizers
  - multiple morphological analyses
- Solution: instead of taking a simple string as input, use a lattice and match all possible paths through the lattice against the training corpus

Generalization Lattice



Segmentation Lattice



Generalizing Language Models

- Use a class-based or template-based model
  - they met last <weekday> to
- Use multiple interpolated models
  - dynamic weighting based on input being translated
    - Use genre classification
    - Compare n-gram statistics of translation candidates to model
    - Compute similarity with associated source-language model

Motivation for Context

- Most EBMT systems treat the training corpus as a bag of examples
- Training corpora are typically sets of documents
- So are the inputs to be translated
- Within a single document, there tends to be a uniformity of usage
- Adjacent sentences will have similar referents for pronouns, etc.
- Therefore, use of context should improve translation quality

Approach to Context

- Looking at three kinds of context:
  - intra-sentential: re-use of a single training example for various fragments of an input sentence
    - multiple fragments from a single training instance will give us increased confidence in the translation
  - inter-sentential: use of training instances near those used for the previous input sentence
    - takes advantage of document-level coherence
  - document-level
    - which training documents look most like the input document?

Intra-Sentential Example

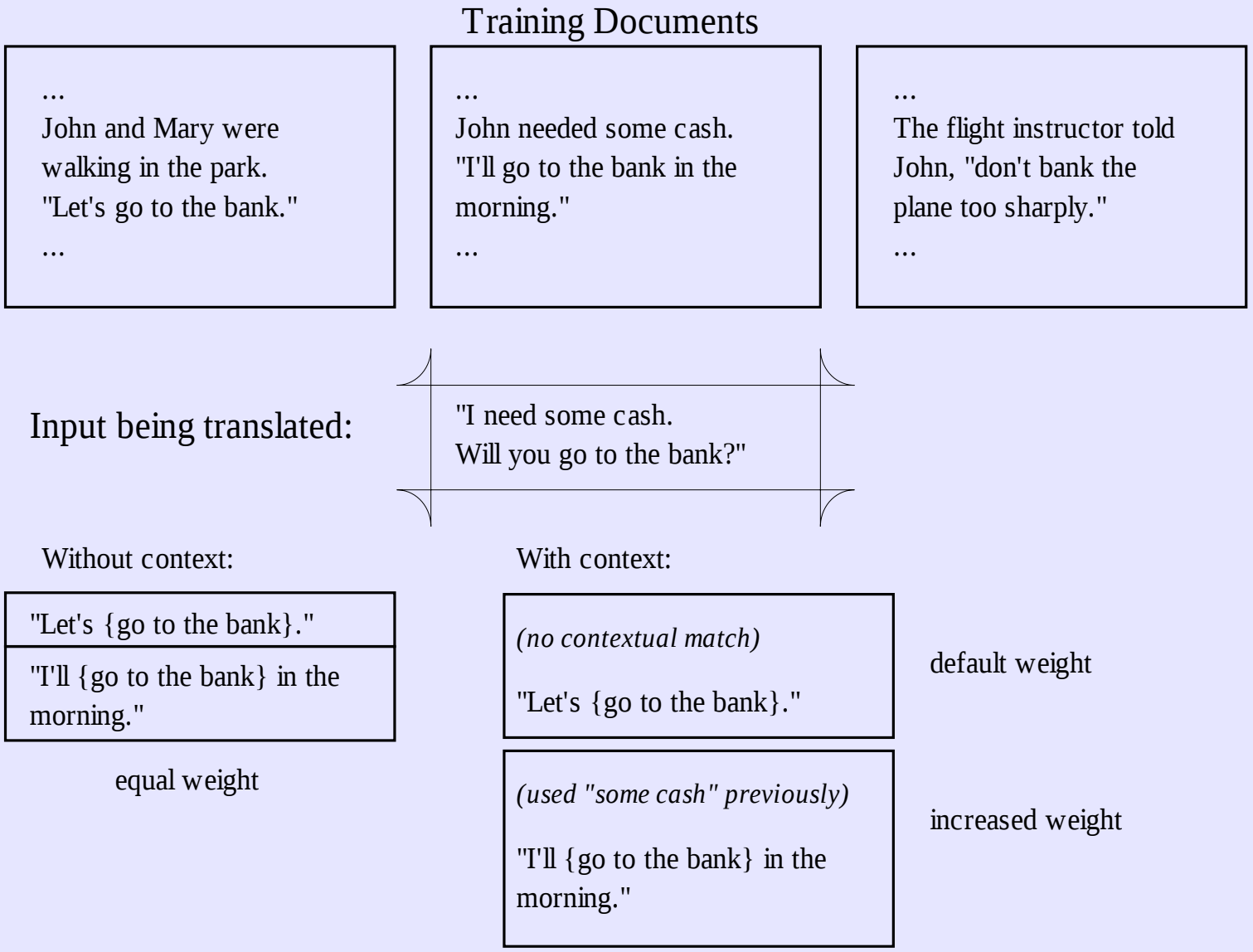
Training Instances

John visited the bank yesterday morning to get some cash.	Bill strolls along the bank every time he comes to the river.

Test Input: John went to the bank to get some cash.

{John} visited {the bank} yesterday morning {to get some cash.}	Bill strolls along {the bank} every time he comes to the river.
bonus for two other matches	default weight

Using Inter-Sentential Context



Intra-Sentential Context

- Matching retrieves multiple examples, and gives a quality score based on the weighted average of the retrieved examples
- Give more weight to training instances that have already been used in translating the current sentence
  - biases scores toward such instances, making them more likely to be selected by the decoder
- Used a greedy approach for ease and efficiency of implementation

Intra-Sentential Context (2)

- Maintain an array of counts, one per instance
- After each match is processed, increment the count for the corresponding training instance
- Adjust weight of the match by current count
- Matches are processed in order by starting offset in the input sentence and reverse order by length
  - matches automatically get a bonus if they are a substring of some other match
  - disjoint matches only receive boost for matches located earlier in the input sentence

Inter-Sentential Context

- Instead of discarding the array of match counts on completing a sentence, make a copy
- Look at match counts not just for active training instance, but also those immediately before and after
- Score bonus is a weighted sum of the counts within five sentence pairs

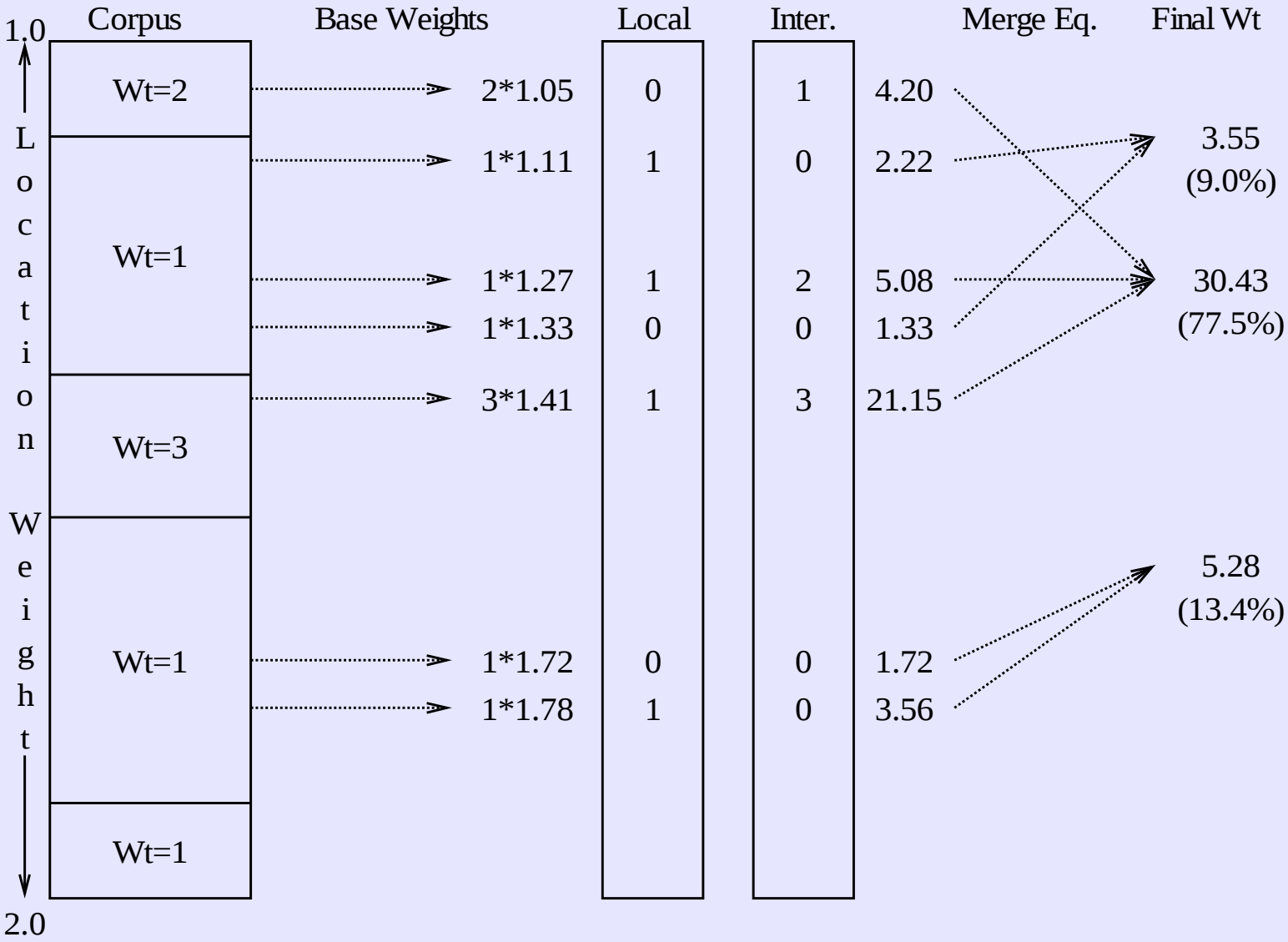
Document-Level Context

- boost weight of training instances in the training documents which are most similar to the input document
- compute similarity using n-gram match statistics
  - ignore n-grams which occur frequently in the corpus
  - uses existing EBMT index lookup code
- each training example is weighted by the normalized match count of its containing document

Integrating Context with Sampling

- impractical to process every match in the corpus
- frequent phrases are subsampled to find 300-400 instances to be processed
  - want to pick the *best* instances
- rank the matched instances by
  - complete match of training example
  - most words of context
  - highest document similarity
- use uniform sampling as a back-off or tie-breaker

Computing Context Bonuses



Generalization in CMU-EBMT

- Basic matching is between strings
  - he **went to several seminars** last month
  - she **went to several seminars** in June
- But those strings need not be surface forms
  - morphological base forms (roots/stems)
    - he [go] to several [seminar] last month
  - equivalence classes
    - he went to several <event-p> last <timespan>
  - templates
    - <PERSON> went to <NP> <TIMESPEC>

Clustering for Generalization

- Too much work to manually create equivalence classes
- Automated clustering methods to the rescue:
  - members of the equivalence class can be used interchangeably, thus appear in similar contexts
  - create a term vector from the words surrounding every instance of a word of interest, then cluster the vectors

Spectral Clustering (1)

- A clustering method that uses a nonlinear dimensionality reduction technique (eigenvalues) on distance matrices
- Can correctly separate non-convex clusters -- even when one completely surrounds another
- Methods exist to automatically determine the correct number of clusters

Spectral Clustering (2)

- Use cosine similarity as the distance metric
- Perform local distance scaling
  - $d(a,b) > d(b,a)$ if *a* has many near neighbors and *b* has few nearby neighbors
- Extract first K eigenvectors, stack them to form a matrix, normalize the matrix (Y), and then perform k-means clustering using each row of Y as a point in K dimensions

Generalizing with Morphology

- Matching on base forms can allow more matches
  - but need ability to filter matches and re-inflect on target side
- Splitting off affixes can improve alignments
  - affixes often correspond to separate words or particles in the other, less-inflected language
- Separating compound words may allow mix-and-match use or generalization of their parts
  - Herzkrankheit → Herz krankheit → {organ}krankheit

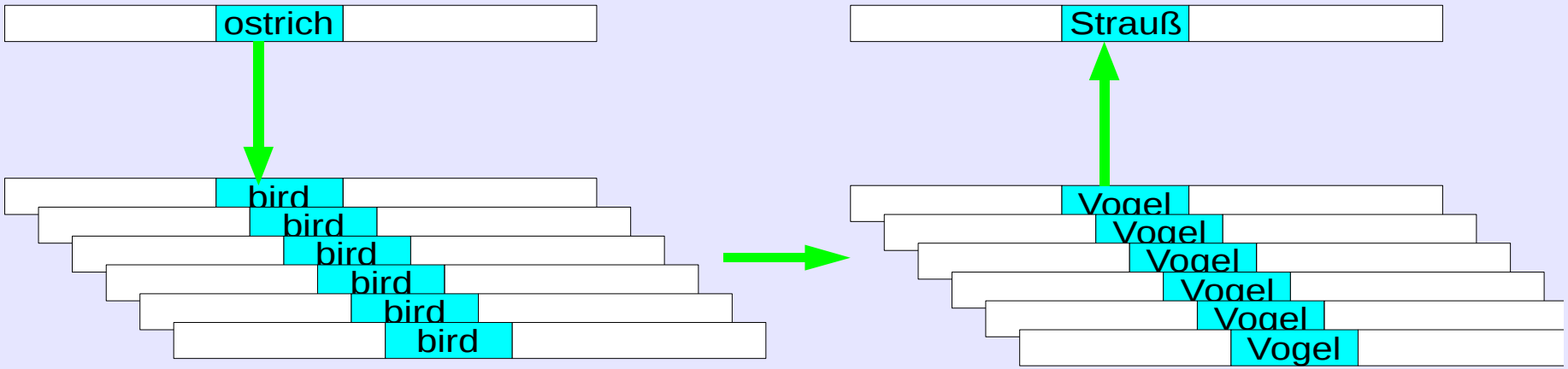
Generalizing with Morphology (2)

- Arabic has rich morphology
  - affixes corresponding to English articles, prepositions, etc.
  - inflectional morphology
  - early performance numbers:
    BLEU 0.20619 → 0.23099 (+11.46%)
  - with more training data and improved system building: 0.45 → 0.47

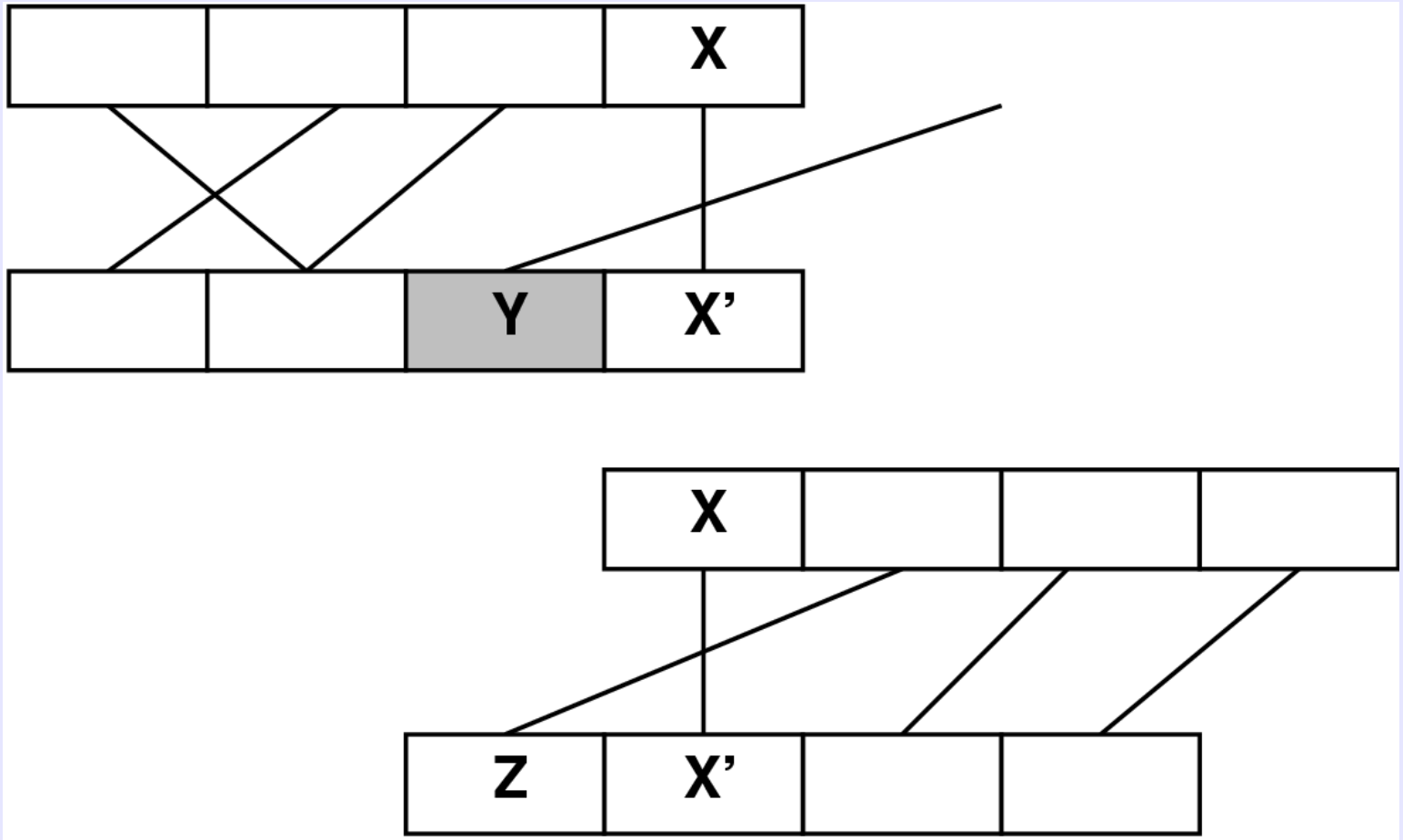
Substitution of Rare and Unknown Words

- Rare words in the input result in few matches of the enclosing phrases
  - poor estimates of translation probabilities, etc.
- Unknown words reduce match lengths
- Replacing such words with common words that occur in the same contexts yields longer matches with more accurate scores

Matching with Replacements



Gap-Filling during Decoding



Convergence of EBMT and SMT

- EBMT systems have been adding more statistics
- SMT systems have become more EBMT-like
  - PBSMT, then dynamic phrase tables
  - Contextual features on phrase table entries
- Cunei is a full hybrid
  - Uses an extension of the SMT formalisms
  - Models every matching training instance separately

Cunei

- Aaron Phillips' dissertation system
- Puts the EBMT approach of using individual training examples into the full SMT formalism
- Open source (www.cunei.org), written in Java

Cunei Formalism

- Standard SMT:
  - $m(s,t,\lambda)=\sum_i \lambda_i \theta_i(s,t)$
- Cunei
  - $m(s,t,\lambda')=\delta+\sum_q \lambda'_q \psi_q$
  - $\psi_q=\frac{\sum_{s',t'} \theta_q(s,s',t',t)\, e^{\sum_i \lambda_i \theta_i(s,s',t',t)}}{\sum_{s',t'} e^{\sum_i \lambda_i \theta_i(s,s',t',t)}}$
- If every training instance is scored equally, the Cunei equation collapses to the SMT equation

Cunei Features

- Defaults to lexical matching, but can index and match at multiple levels
  - e.g. POS, morphological tags
- Supports gapped matches
  - Approximate, uses subsampled list of phrasal matches on either side of the gap
- Uses a gradient-descent parameter optimizer instead of MERT because of the large number of parameters it supports

Summary

- A family of corpus-based approaches to MT which use individual training examples at run time
- Three phases to translation: matching, transfer/adaptation, and recombination
- Matches can be lexical, generalized, or use a deeper representation such as parse trees
- Matches can be retrieved or weighted by quality of match against the input *and its surrounding context*
- EBMT and SMT are converging