Translation Memories

Timothy Baldwin

CSLI, Stanford University

tbaldwin@csli.stanford.edu

9th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2002), Keihanna, Japan.

Structure of this Tutorial

- **Part I:** A basic introduction to translation memories and translation retrieval
- **Part II:** Case study of Japanese–English translation retrieval
- **Part III:** Enhancements and extensions to the basic paradigm

PART I: A Basic Introduction

The Reality of Translation

- The bulk of translation work relates to technical materials (controlled languages = contracts, user manuals, field reports, etc.) NOT literary prose
- Technical translation calls for in- and inter-document consistency (terminological, structural, phrasal, etc.) and translation accuracy
- It is timely and expensive to train technical translators
- Technical translation is monotonous!

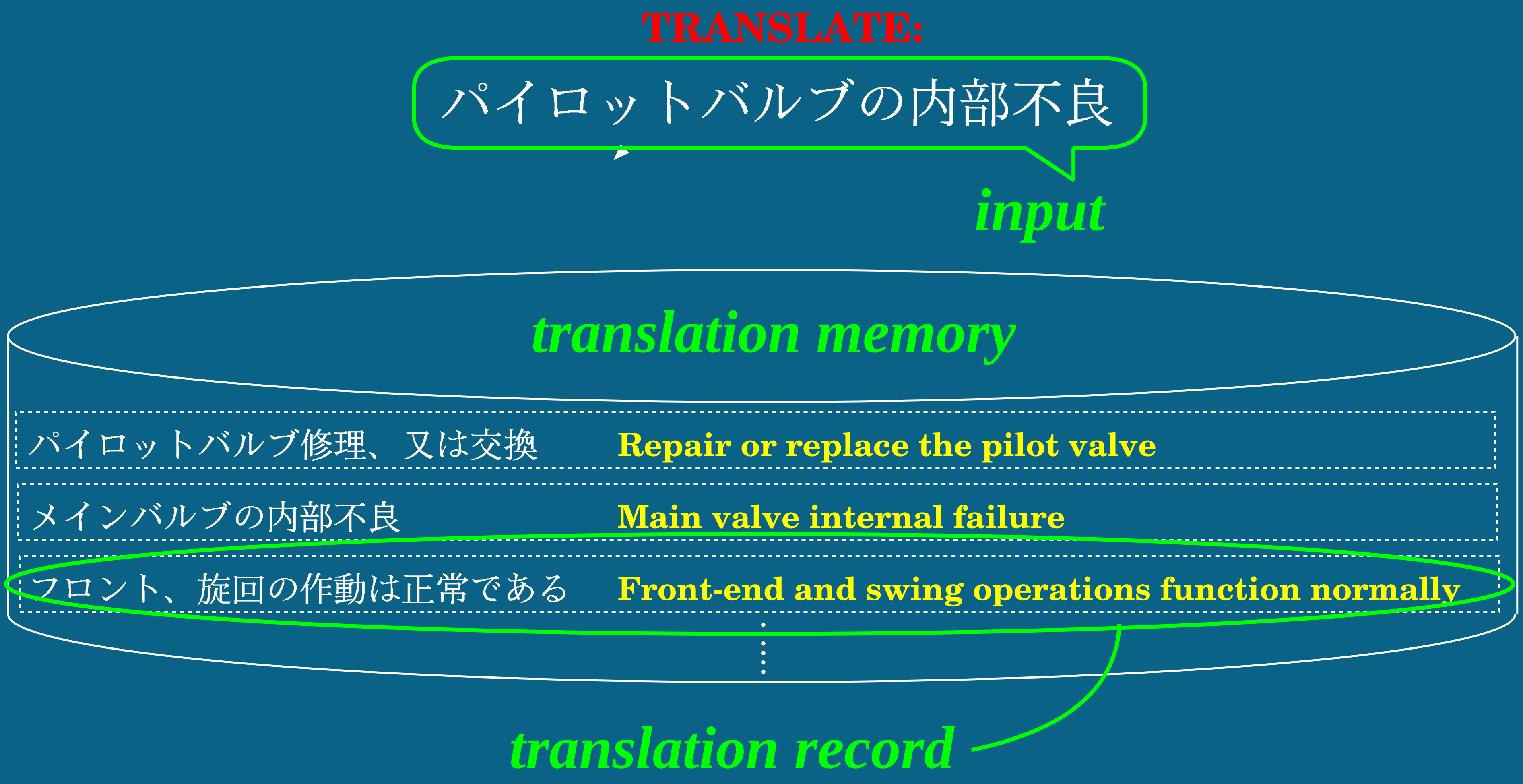
Enter the Computer!

- Computers are consistent (although possibly consistently *wrong*!)
- Computers have an infinite boredom tolerance
- Computers are good at performing tasks they know how to do (e.g. translate a given string based on a human translation)
- BUT computers (MT systems) are temperamental at best at translating novel strings

TRANSLATE:

パイロットバルブの内部不良

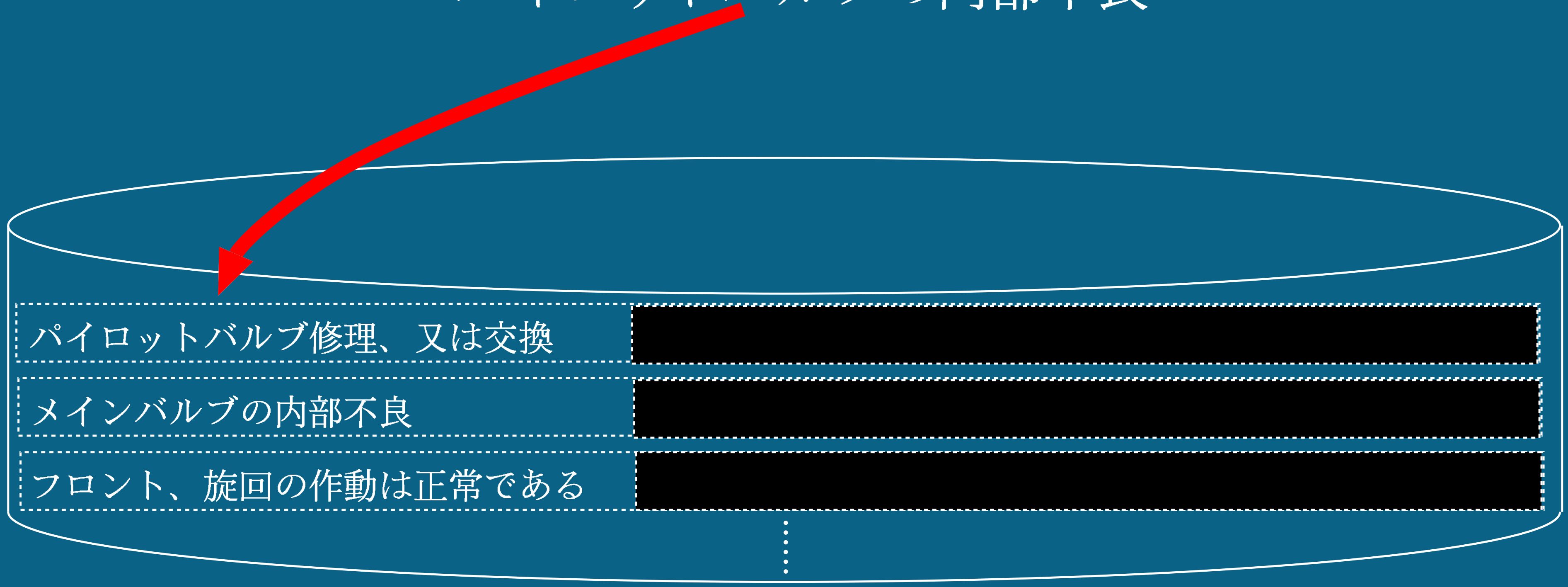
パイロットバルブ修理、又は交換	**Repair or replace the pilot valve**
メインバルブの内部不良	**Main valve internal failure**
フロント、旋回の作動は正常である	**Front-end and swing operations function normally**
⋮	



パイロットバルブ修理、又は交換	**Repair or replace the pilot valve**
メインバルブの内部不良	**Main valve internal failure**
フロント、旋回の作動は正常である	**Front-end and swing operations function normally**

TRANSLATE:

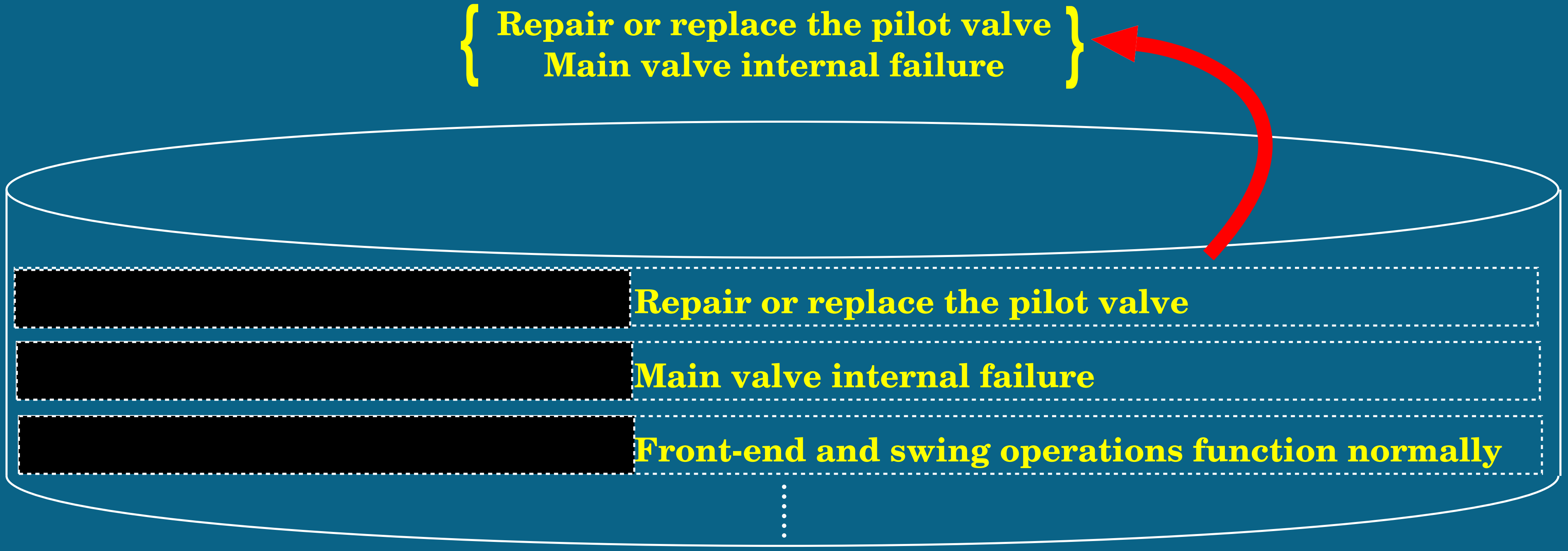
パイロットバルブの内部不良



パイロットバルブ修理、又は交換	[redacted]
メインバルブの内部不良	[redacted]
フロント、旋回の作動は正常である	[redacted]

TRANSLATE:

パイロットバルブの内部不良

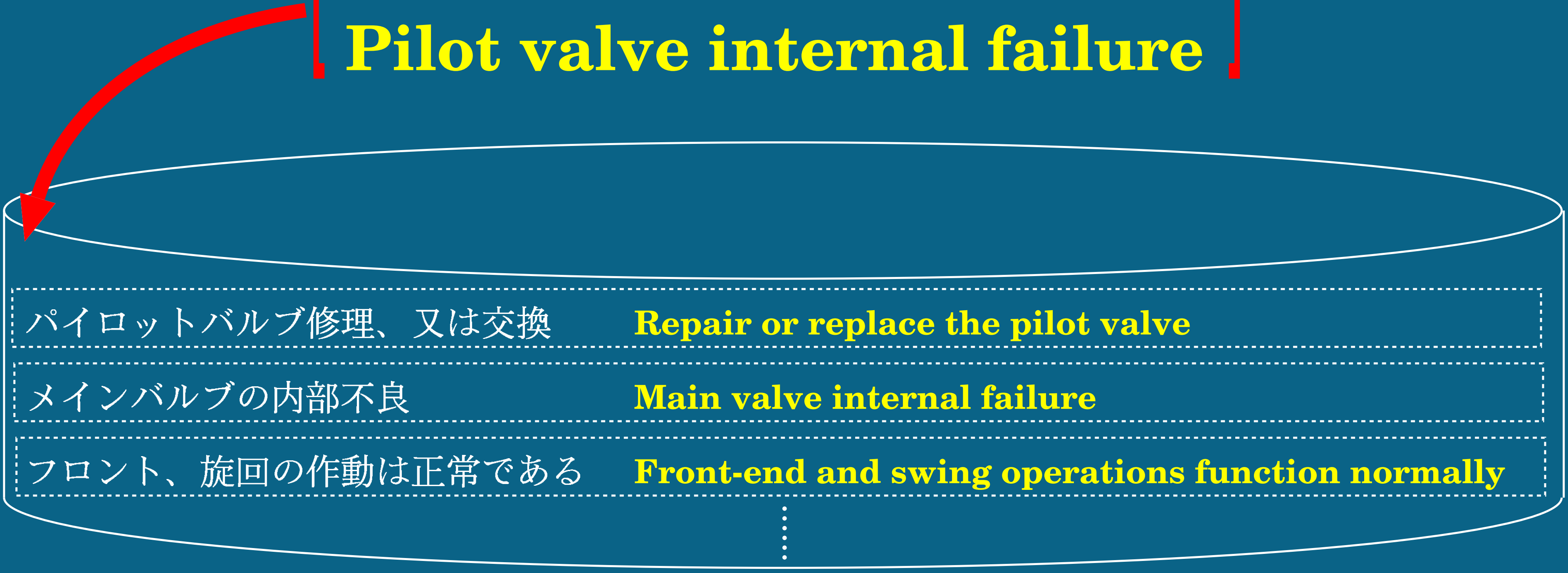


{ Repair or replace the pilot valve
Main valve internal failure }

[redacted]	Repair or replace the pilot valve
[redacted]	Main valve internal failure
[redacted]	Front-end and swing operations function normally
⋮	

TRANSLATE:

[パイロットバルブの内部不良
Pilot valve internal failure]



パイロットバルブ修理、又は交換	**Repair or replace the pilot valve**
メインバルブの内部不良	**Main valve internal failure**
フロント、旋回の作動は正常である	**Front-end and swing operations function normally**
⋮	

Key Terms

- **Translation record:** source language (L1) string coupled with its target language (L2) translation
- **Translation memory (TM):** database of translation records
- **Translation retrieval (TR):** the process of retrieving translation(s) from the translation memory based on L1 similarity with the input

Human–computer Symbiosis (1)

- The role of the computer:
  - ⋆ translate previously-seen inputs (by way of previously-generated translations)
  - ⋆ suggest translations for inputs similar to previously-seen inputs
  - ⋆ check on the consistency of the translation (based on past translation data)
  - ⋆ streamline the translation process so as to minimise translator effort

Human–computer Symbiosis (2)

- The role of the human:
  - ⋆ concentrate on novel data
  - ⋆ post-edit any computer-generated translations
  - ⋆ post-edit the final document to ensure readability, factual accuracy

Translation Memory Feedback Loop

1. TM system suggests translation candidates for given input based on best L1 matches in translation memory
2. User fashions translation for current input with or without a translation candidate from the TM
3. TM system feeds the final translation (and original input) back into the translation memory for use in subsequent retrievals

Key Assumptions

- **Uniqueness of translation:** a given L1 string has a unique optimal translation in the given domain
- **Lexico-semantic similarity:** for strings S and T, $sim_{lex}(S,T) \propto sim_{sem}(S,T)$
- **Cross-lingual similarity:** for translation records $\langle i,j\rangle$ and $\langle x,y\rangle$, $sim_{SL}(i,x) \propto sim_{TL}(j,y)$

Optional Assumption

- **Discourse independence:** an optimal document-level translation can be acquired by concatenating L2 units from the translation memory in the same order as the original L1 units

HCI-related Desiderata for TM Systems

- Speed: the user shouldn't be kept waiting for TM output
- Invisibility: the only interaction the user should have with the TM system is in choosing whether or not to recycle previous translation data in translating a given string
- Seamlessness: the user should not have to make any sacrifices in terms of text processing functionality in order to use a TM system

Translation Retrieval and Example-based Machine Translation

- Translation retrieval first step of EBMT:
  - ⋆ TM systems simply return a set of translations to the user
  - ⋆ EBMT systems take the translation fragments and assemble them into an overall translation for the original input
- Clearer-defined feedback loop in TM systems (EBMT systems tend not to trust their own output!)

Translation Retrieval and Information Retrieval (IR)

- Input and translation records directly comparable in TR; query and documents heterogeneous in IR $\longrightarrow$
  - ⋆ retain original structure of input with TR, but abstract away from linguistic structure of query in IR
  - ⋆ relative length/segment overlap of input and translation records important in TR, not in IR

PART II: Case Study of Japanese–English Translation Retrieval

Background to TR research

- Traditional work on translation retrieval has tended to neglect cross-system evaluation
- Little justification/validation of decisions made in the design of translation retrieval methods
- Implicit assumption that greater linguistic stringency leads to better results
- TR considered to be a stable technology, which has reached the limits of its technological potential

Fundamental Questions

- What basic decisions do people make in designing a translation retrieval method?
- What is the parameter space?
- What effect do the different parameters have on retrieval performance?
- Use the Japanese–English translation retrieval task as a common platform in attempting to answer some of these questions

Orthogonal Parameters in Translation Retrieval (1)

- **Segment granularity**: unit character vs. word segments (character- vs. word-based indexing)
- **Segment order**: segment order-oblivion (bag-of-words approach) vs. segment order-sensitivity
- **Segment contiguity**: segment contiguity oblivion vs. segment contiguity awareness

Orthogonal Parameters in Translation Retrieval (2)

- **Segment weighting**: the effect of weighting different segment types differently
- **Thresholds on translation utility**: the effect of different numerical thresholds over string similarities, on the utility of the resultant translation candidates
- Also: **partition granularity**

Underlying question

- What is the cost of the different methods, and is the resultant accuracy gain justified?
- Do more computationally expensive methods produce commensurate retrieval accuracy gains?

Methodology (1)

- Evaluate the relative retrieval performance of each parameter setting:

1. Morpheme- vs. character-segmented strings
2. Match up a selection of well-accepted bag-of-words methods against a selection of segment order-sensitive methods
3. Model local segment contiguity with N-grams (single-order and mixed models), and adjust the N-gram order

Methodology (2)

- Test various segment weighting methods (static and dynamic)
- Suggest method to justify translation utility threshold values

Segmentation Modules Tested

- ChaSen (NAIST)
- JUMAN (Kyoto Uni.)
- ALTJAWS (NTT CS Labs.)

Segment Weighting

- Not all segments are equally reflective of semantic similarity between strings (e.g. stop vs. content words)
- Describe the lexico-semantic import of each segment s by way of the weight $sweight(s)$
- Static vs. dynamic segment weighting methods:
  - ⋆ static methods easier to implement, update
  - ⋆ dynamic methods domain-tunable

Weighting Methods

- Static weighting:
  - ⋆ **character type** (kanji vs. kana)
  - ⋆ **word type** (stop vs. content word)
  - ⋆ **segment length** (in characters)
- Dynamic weighting: e.g. **IDF**

$$idf(s) = \min\left(\log\frac{|TM|+1}{|\{tr : tr \in TM, tr \ni s\}|}, max\right) \tag{1}$$

Weightings Assumed Here:

English *sweight* Schema:

Segment type	*sweight*
punctuation/stop words	0
other words	1

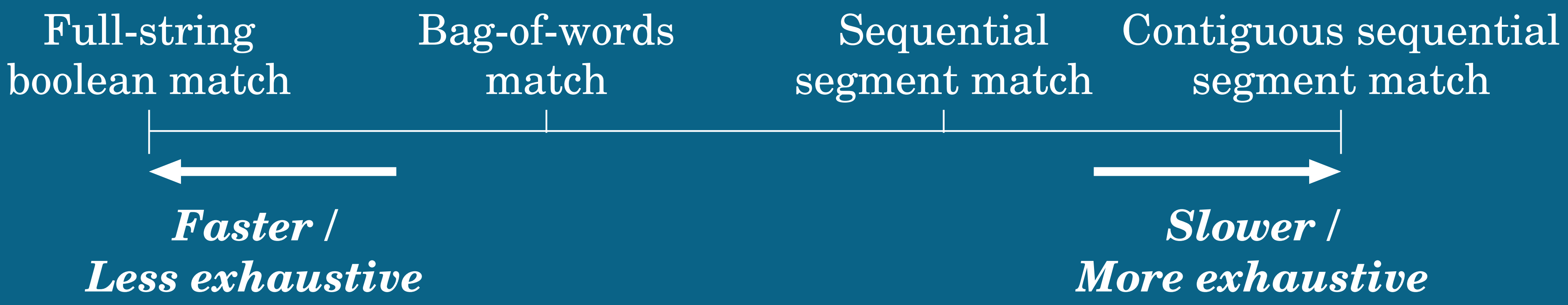
Japanese *sweight* Schema:

Segment type	*sweight*
punctuation	0
other segments	1

String Comparison Methods

- **Bag-of-words methods**:
  - ⋆ Vector space model
  - ⋆ Token intersection
- **Segment order-sensitive methods** (DP implementation):
  - ⋆ 3-operation edit distance
  - ⋆ 3-operation edit similarity
  - ⋆ Weighted sequential correspondence

The Continuum of Match Exhaustivity vs. Speed


Full-string boolean match
Bag-of-words match
Sequential segment match
Contiguous sequential segment match
Faster / Less exhaustive
Slower / More exhaustive

Example Strings

(1) 冬·の·雨 [$huyu{\cdot}no{\cdot}ame$] “winter rain”

(2a) 夏·の·雨[$natu{\cdot}no{\cdot}ame$] “summer rain”

(2b) 雨·の·夏 [$ame{\cdot}no{\cdot}natu$] “a rainy summer”

(2c) 雨·の·冬 [$ame{\cdot}no{\cdot}huyu$] “a rainy winter”

(2d) 真·冬·の·雨 [$ma{\cdot}huyu{\cdot}no{\cdot}ame$] “ mid-winter rain”

Vector Space Model

- For vectors $\vec{S}$ and $\vec{T}$ of segment types in strings S and T, respectively:

$$\cos(\vec{S},\vec{T}) = \frac{\sum_j sweight(seg_j)s_j t_j}{\sqrt{\sum_j sweight(seg_j)s_j^2}\sqrt{\sum_j sweight(seg_j)t_j^2}}$$

- Segment order-insensitive similarity

- Best match for (2c): $\frac{1\times1+1\times1+1\times1}{\sqrt{3}\sqrt{3}} = \frac{3}{3} = 1$ ◀

Token Intersection (=Dice's Coefficient)

- For strings S and T of (weighted) segment length $wlen(S)$ and $wlen(T)$, respectively:

$$tint(S,T) = \frac{2 \times \sum_{e \in S,T} sweight(e) \min \left(freq_S(e), freq_T(e) \right)}{wlen(S) + wlen(T)}$$

- Segment order-insensitive similarity
- Best match for (2c): $\frac{1\times1+1\times1+1\times1}{\sqrt{3}\sqrt{3}} = \frac{3}{3} = 1$ ◀

3-operation Edit Distance

- Distance between strings in terms of the (weighted) number of segment insertions and deletions (not substitutions) required to transform one into the other
- Segment order-sensitive *distance*
- Best match for (2d): $D_{3op}((1),(2d)) = 1$ ◀

$$D_{3op}(S,T) \quad = \quad d_3(m,n)$$

$$d_3(i,j) = \begin{cases} 0 & \mathit{if}\ \ i = 0 \wedge j = 0 \\ d_3(0, j-1) + \mathit{sweight}(t_j) & \mathit{if}\ \ i = 0 \wedge j \neq 0 \\ d_3(i-1, 0) + \mathit{sweight}(s_i) & \mathit{if}\ \ i \neq 0 \wedge j = 0 \\ \min \begin{pmatrix} d_3(i-1, j) + \mathit{sweight}(s_i), \\ d_3(i, j-1) + \mathit{sweight}(t_j), \\ m_3(i,j) \end{pmatrix} & \mathit{otherwise} \end{cases}$$

$$m_3(i,j) = \begin{cases} d_3(i-1, j-1) & \mathit{if}\ \ s_i = s_j \\ \infty & \mathit{otherwise} \end{cases}$$

3-operation Edit Similarity

- Normalise edit distance $S_{3op}(S,T)$ according to weighted lengths of the strings S and T:

$$S_{3op}(S,T) \;=\; 1 - \frac{D_{3op}(S,T)}{wlen(S) + wlen(T)}$$

- Segment-order sensitive similarity
- Best match for (2d): $S_{3op}((1),(2d)) = \frac{6}{7}$ ◀

Weighted Sequential Correspondence

- Degree of contiguous substring similarity between strings, weighted according to the contiguity of segment matches (and $sweight$)
- Assume $Max = 3$
- Segment order/contiguity-sensitive similarity
- Best match for (2d): $wseq^*((1),(2\text{d})) = \frac{2\times 6}{6+9} = \frac{4}{5}$ ◀

$$S_w(S,T) \quad = \quad s(m,n)$$

$$s(i,j) = \begin{cases} 0 & \textit{if } i = 0 \vee j = 0 \\ \max\begin{pmatrix} s(i-1,j), \\ s(i,j-1), \\ s(i-1,j-1) + m_w(i,j) \end{pmatrix} & \textit{otherwise} \end{cases}$$

$$m_w(i,j) = \begin{cases} cm(i,j) \times sweight(i) & \textit{if } s_i = s_j \\ 0 & \textit{otherwise} \end{cases}$$

$$cm(i,j) = \begin{cases} 0 & \textit{if } i = 0 \vee j = 0 \vee s_i \neq t_j \\ \min(Max, cm(i-1,j-1)+1) & \textit{otherwise} \end{cases}$$

$$wseq^*(S,T) = \frac{2 \times S_w(S,T)}{S_w(S,S) + S_w(T,T)}$$

N-gram Models Tested

- Complement string comparison methods with model of local segment contiguity (which preserves basic segment order):
  - ⋆ **unigrams**: バルブ = バ·ル·ブ
  - ⋆ **bigrams**: バルブ = バル·ルブ
  - ⋆ **mixed unigrams/bigrams**: バルブ = バ·バル·ル·ルブ·ブ
  - ⋆ (Trigrams, mixed bigrams/trigrams)

Pruning the Search Space

- Use inverted file to search only over those translation records which overlap in segment content with the input
- Sort overlapping translation records in decreasing order of segment overlap, apply halting conditions over string length
- For segment order-sensitive methods, use dynamic bounds on string length to hone in on those translation records which have a chance of bettering the best match to date

Halting Conditions

Comparison method	*Halting condition*
3-op edit distance	$\beta > \alpha$
3-op edit similarity	$1 - \frac{\beta}{wlen(IN)+wlen(TM_i)} < \alpha$
Weighted seq corr	$\frac{2\beta}{wlen(IN)+wlen(TM_i)} < \alpha$

$\beta =$ segment overlap between IN and TM_i

$\alpha =$ current best matching score

Dynamic Bounds on String Length

Comparison method	*Upper bound on length*	*Lower bound on length*
3-op edit distance	$wlen(IN) + \alpha$	$wlen(IN) - \alpha$
3-op edit similarity	$\frac{(2-\alpha)wlen(IN)}{\alpha}$	$\frac{\alpha wlen(IN)}{2-\alpha}$
Weighted seq corr	$\frac{(2-\alpha)wlen(IN)}{\alpha}$	$\frac{\alpha wlen(IN)}{2-\alpha}$

$\beta =$ segment overlap between IN and TM_i

$\alpha =$ current best matching score

Evaluation Datasets

- **Dataset 1**: 3,033 translation records
  - ⋆ from technical field reports
  - ⋆ ave. translation record length = 27.7 characters (J)/13.3 words (E)
- **Dataset 2**: 61,236 translation records
  - ⋆ from the JEIDA parallel corpus
  - ⋆ ave. translation record length = 76.3 characters (J)/35.7 words (E)

Dataset 1 – Technical Field Reports

i. モニタから水温センサ間のハーネス断線 = break in the harness from the monitor to the coolant temperature sensor

ii. 又、午前中バッテリ交換した後夕方まで休車させるとバッテリが完全放電してしまった。 = additionally , after having replaced the battery in the morning and left the machine idle for the day , the battery is flat by evening

iii. 点検しハーネスを規定途通りに接続した。= inspected and corrected the harness connection

iv. fm/am ラジオのチャンネルと時計のメモリーがキースイッチoffにすると消える。 = the channel setting of the am / fm radio and clock memory are deleted when the ignition is turned off

⋮

Dataset 2 – Technical Field Reports

i. 最近10年間の民間企業の研究者数の推移についてみてみると，研究本務者は非常に安定して増加を続けてきている（第2-1-25図）。 = the number of regular researchers has shown a steady increase over the last ten years (figure 2-1-25)

ii. 農林水産省においては,「新需要創出のための生物機能の開発・利用技術の開発に関する総合研究」の一環として，バイオプラスチック・シルクレザー等の生物素材に係る研究開発を実施している。 = the ministry of agriculture forestry and fisheries is promoting research and development on bio-materials such as bio-plastics and silk leather as part of its integrated research program for effective use of biological activities to create new demand

iii. このように生産面からみると今般の景気回復基調がいわゆる「半導体景気」という性格も持っていることが分かる。 = in this way , in terms of production the underlying tone of the latest economic recovery has a semiconductor economy character

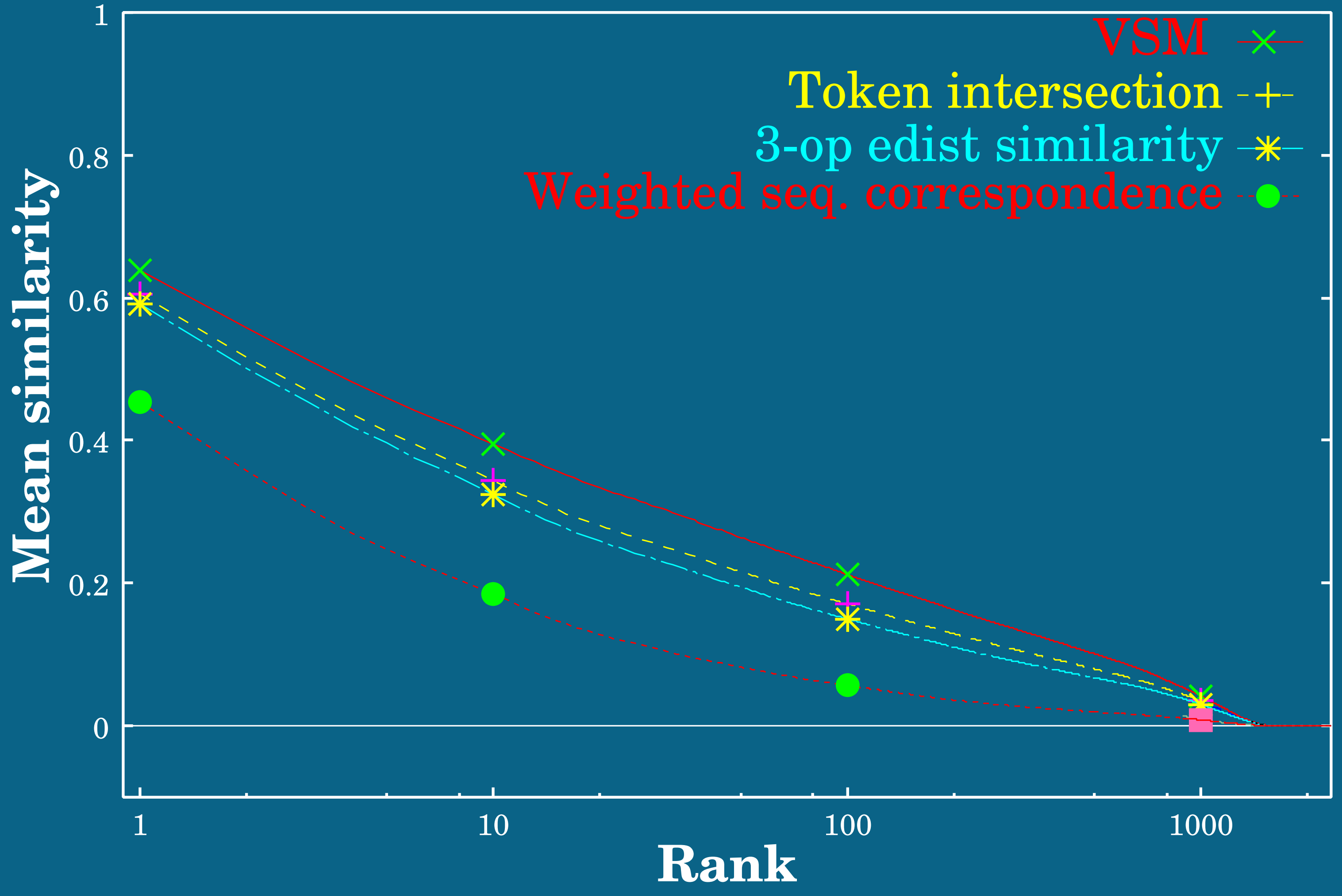
⋮

Thresholds on Retrieval Utility

- Varying level of translation utility according to the score associated with the output
- Better to produce no output than unusable output
- Apply threshold on best similarity/distance value; in case not met, return empty string

String comparison method	*Threshold*
Vector space model	0.5
Token intersection	0.4
3-operation edit distance	$wlen(IN)$
3-operation edit similarity	0.4
Weighted seq. correspondence	0.2

Similarity Depreciation



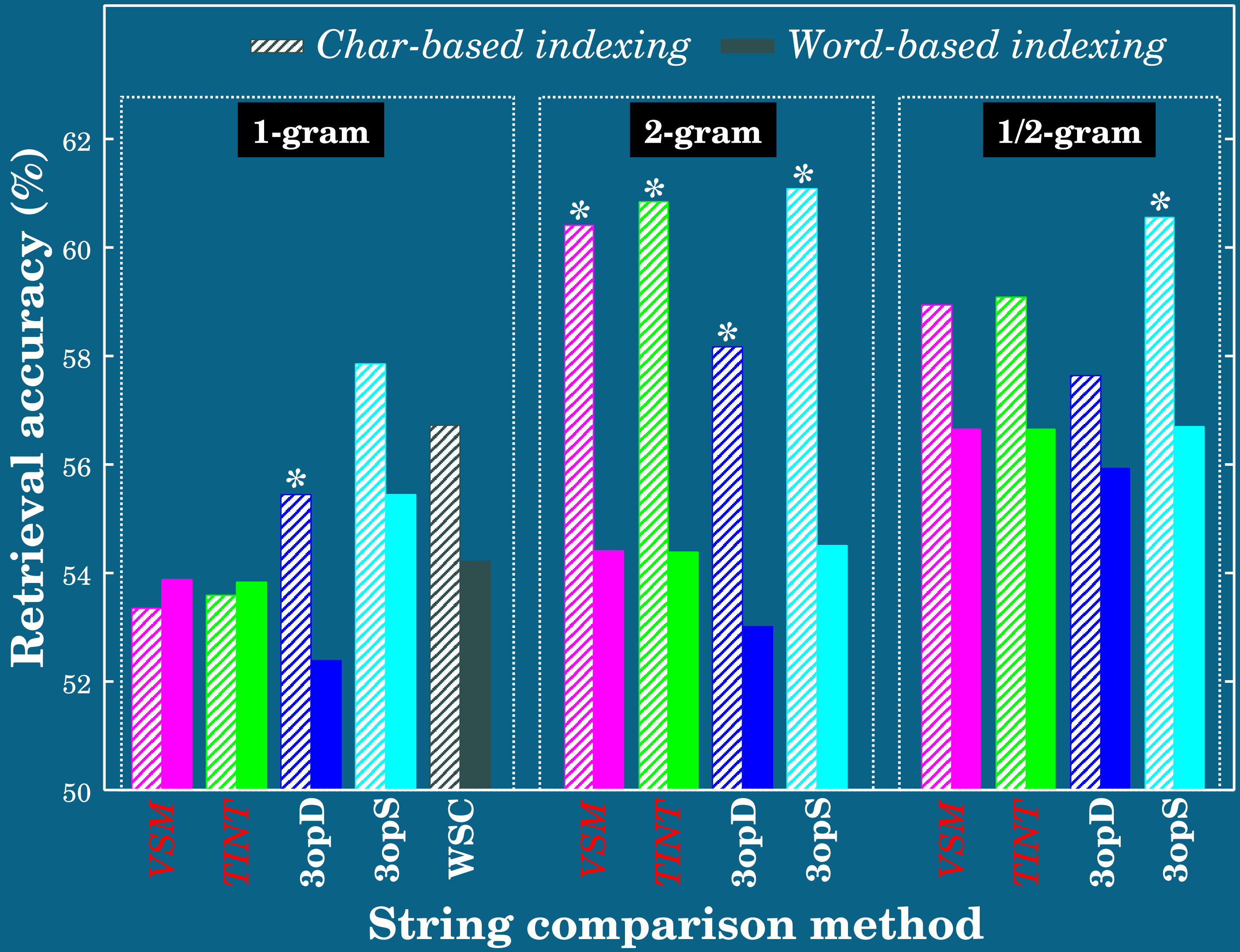
Evaluation Procedure

- Use *10-fold semi-stratified cross-validation* to determine retrieval accuracy
- On each iteration of cross-validation, take the L1 component of the test data as the set of inputs, and the L2 translations as the model translations
- Use an arbitrary string comparison method to determine the set of "optimal translations" for a given input, as the translations in the TM most similar/least displaced from the model translation

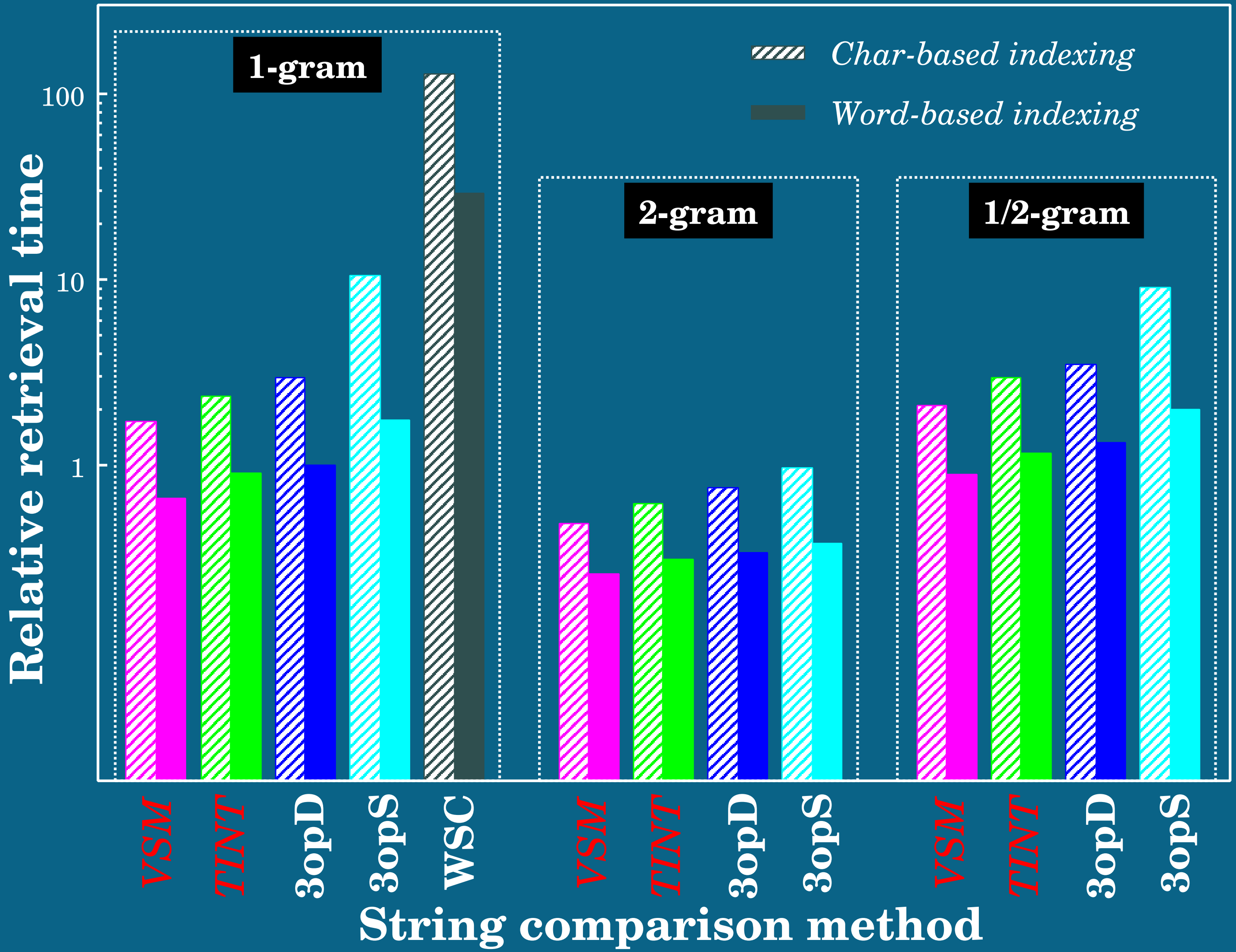
- "Retrieval accuracy" defined as the proportion of inputs for which an optimal translation was retrieved

- Retrieval accuracy averaged over the results from the 3-op edit distance and weighted sequential correspondence methods

Basic Retrieval Accuracies



Basic Retrieval Times



Preliminary Findings

- Char-based indexing is superior to word-based indexing, particularly when combined with bigram and mixed bigram segment contiguity models
- Bag-of-words and segment order-sensitive methods are roughly equivalent in retrieval accuracy, but bag-of-words methods are faster
- Explicit modelling of local segment contiguity does enhance retrieval performance, with bigrams being the

best N-gram choice for char-based indexing (both more accurate and faster), and mixed unigrams/bigrams the best choice for word-based indexing

- The best string comparison method is 3-op edit similarity (marginally)

Questions to Come out of this

- Is the superiority of char-based indexing:

(a) a universal trait or particular to the original dataset?
(b) particular to ChaSen or observable for other segmenters also?

- Equivalence of the bag-of-words and segment order-sensitive methods particular to the original dataset?

- How does variation in the size of the dataset affect retrieval performance?

Segmentation and Retrieval Accuracy

- Redo retrieval evaluation with data segmented with:

(a) JUMAN
(b) ALTJAWS (−lexical normalisation)
(c) ALTJAWS (+lexical normalisation)

- Lexical normalisation: convert all morphemes into canonical form, e.g. 行っ|た → 行く|た, 成り行き → 成行き, 充分 → 十分

Evaluation of Segmentation Performance

$$\text{Segment precision} = \frac{\#\text{ correct segments in system output}}{\text{Total }\#\text{ segments in system output}}$$

$$\text{Segment recall} = \frac{\#\text{ correct segments in system output}}{\text{Total }\#\text{ segments in correct analysis}}$$

$$\text{Sentence accuracy} = \frac{\#\text{ sentences containing no segmentation errors}}{\text{Total }\#\text{ sentences}}$$

$$\text{Total segment types} = \#\text{ segment types in overall data}$$

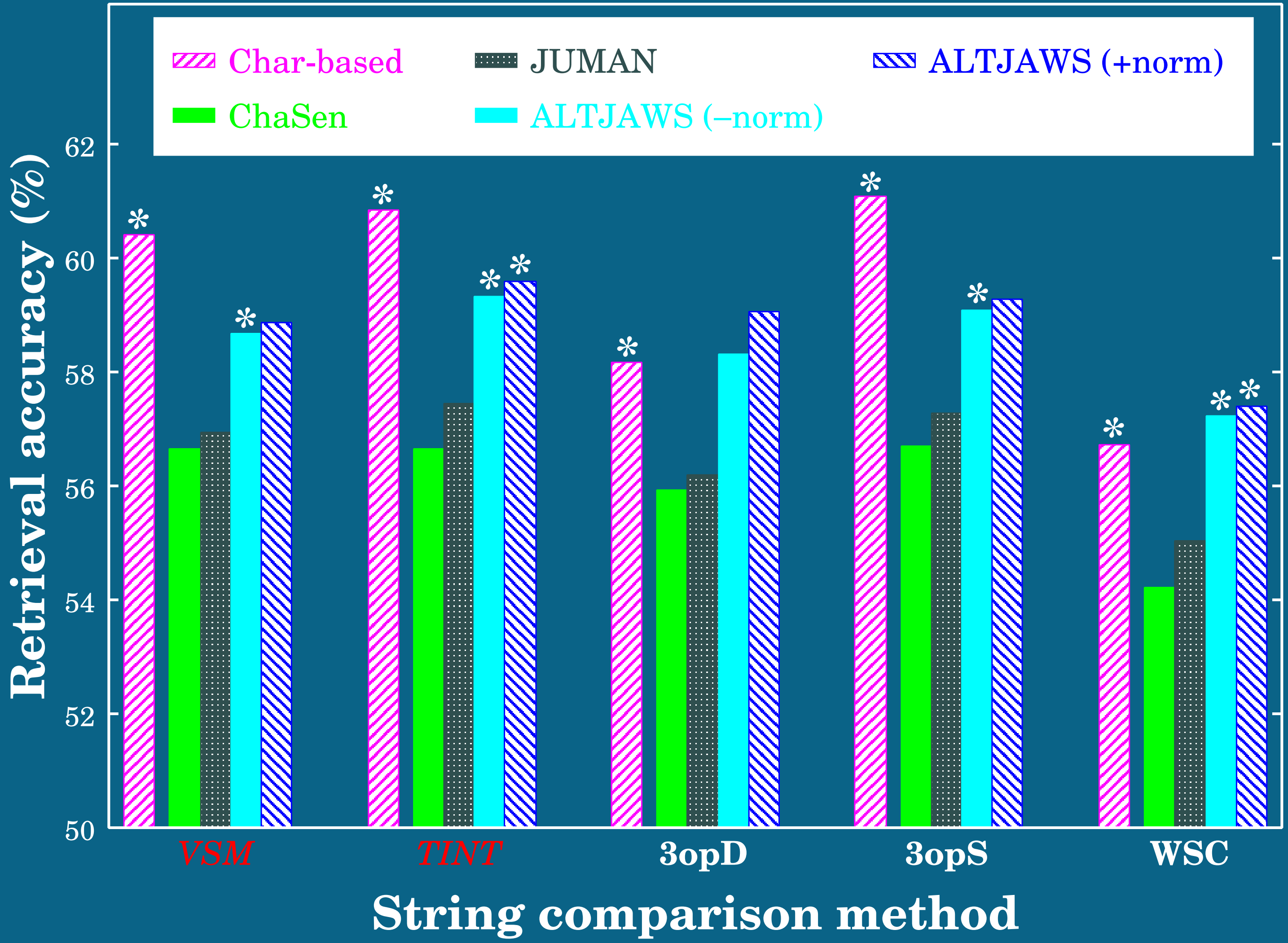
Expectations

- Higher segment precision and recall → higher retrieval accuracy
- Less segment tokens → greater segmentation consistency → higher retrieval accuracy
- Slight complication with different segment granularities of different systems (e.g. していた → し|て|いた or して|いた or していた)

Segmentation Performance

	ChaSen	JUMAN	ALTJAWS
Segs./translation record	13.0	12.0	11.7
Segment precision	98.3%	98.3%	98.6%
Segment recall	98.1%	96.2%	97.7%
Sentence accuracy	70.5%	59.0%	72.0%
Total segment types	650	656	634

Results for Different Segmenters



Combined Segmentation and Translation Performance

- ChaSen best performer in terms of segment precision and recall, but highly inconsistent in cases of error
- ALTJAWS produces more errors but is consistent in doing so
- Greater consistency, and robustness over key terms (katakana words) correlates with higher retrieval accuracy

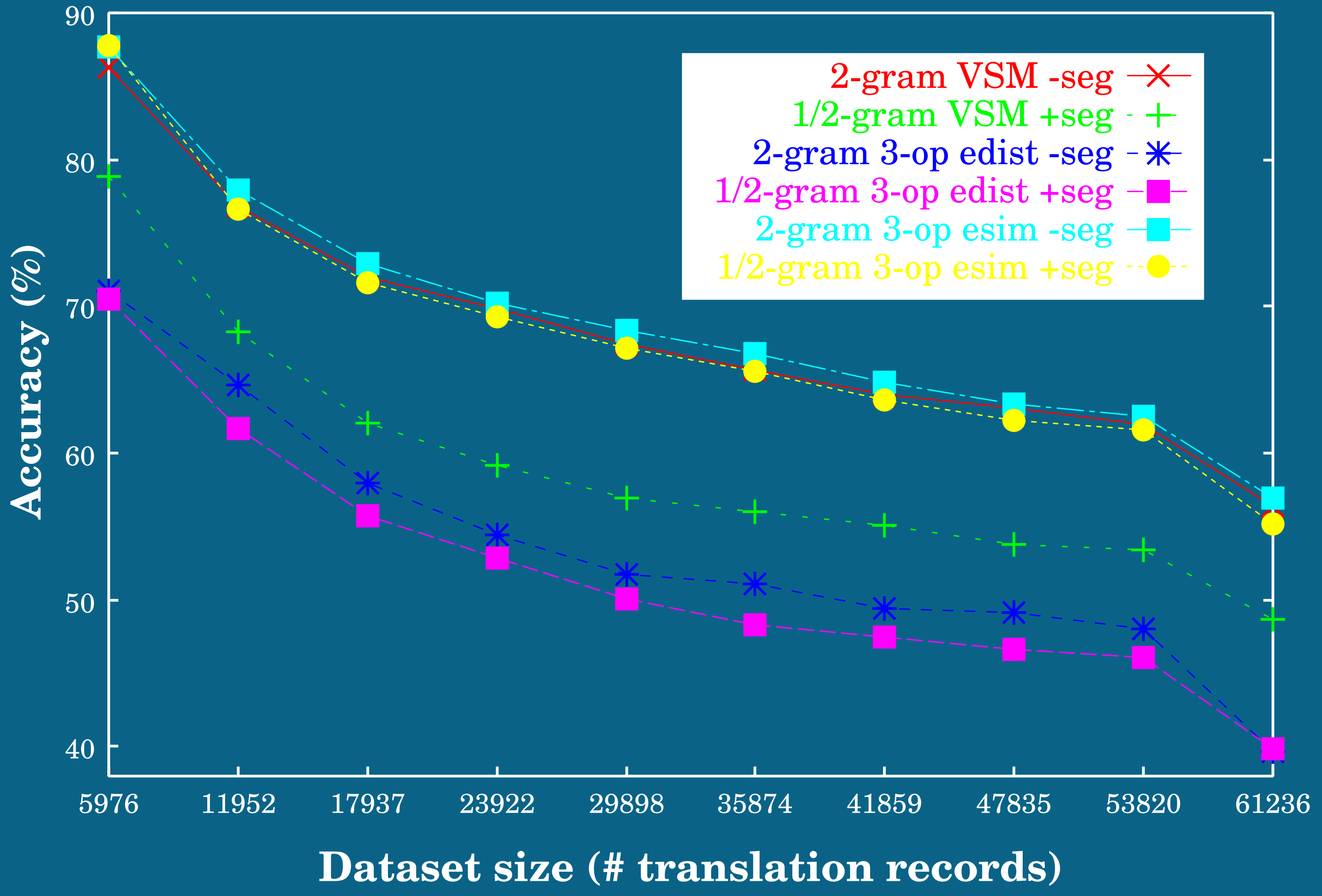
Preliminary Findings

- JUMAN slightly better than ChaSen, ALTJAWS better again
- Lexical normalisation produces only a slight gain (due to the relative infrequency of lexical alternation)
- Char-based indexing still superior in terms of accuracy for best-performing methods (+speed gains)

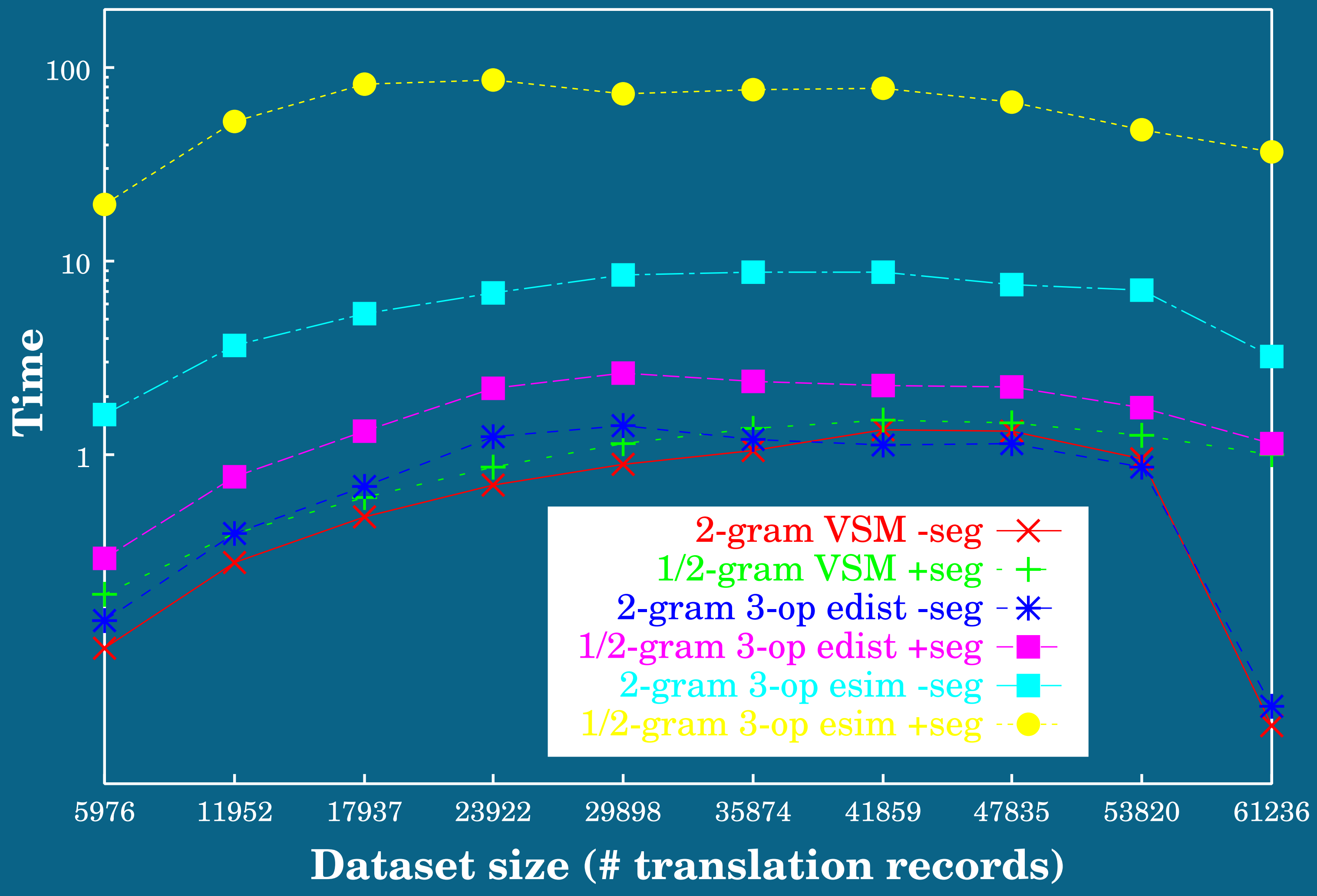
Scalability of Performance

- **Limitation of evaluation to here**: No sense of whether the findings to date are particular to the given dataset or data size
- **Solution**: Take a larger dataset (JEIDA parallel corpus), and evaluate retrieval performance over data increments of increasing size
- Compare the relative accuracy and speed of the vector space model, 3-op edit distance and 3-op edit similarity, under the two indexing paradigms

Retrieval Accuracy over Different TM Sizes



Retrieval Time over Different TM Sizes



Summary of Results for Performance Scalability

- For all methods tested, the absolute disparity between character- and word-based indexing was maintained over different data sizes (character-based indexing superior to different degrees)

- 3-operation edit similarity and the vector space model were found to be superior and perform at almost identical levels of accuracy under character-based indexing (bag-of-words $\equiv$ segment order-sensitive string comparison)

- 3-operation edit similarity ran about 10 times slower than the other two methods for char-based indexing, and about 100 times slower for word-based indexing

Qualitative Evaluation

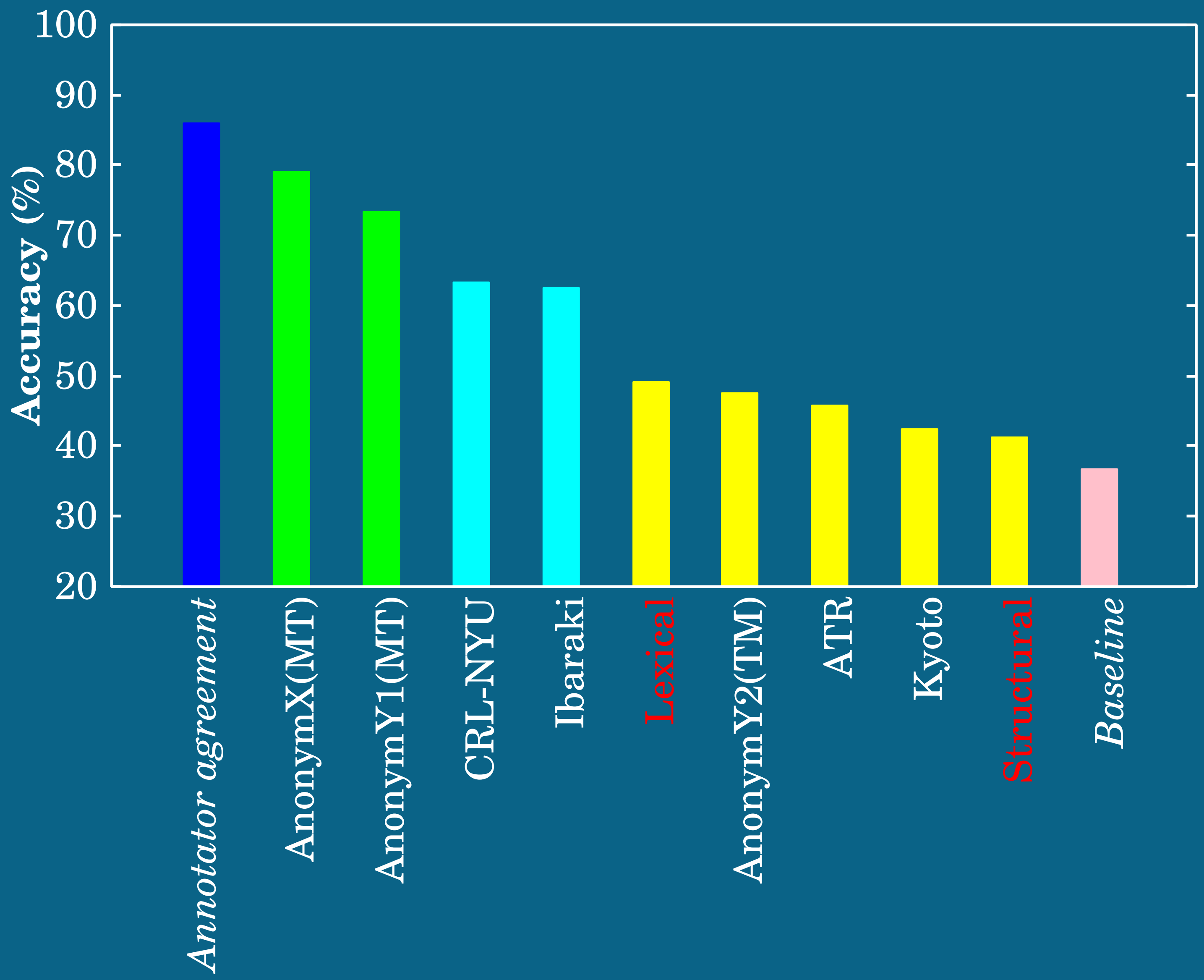
- **Character- vs. word-based indexing**
  - ⋆ longer matching strings scored higher under character-based indexing (picks up on katakana sequences = technical terms)
- **Bag-of-words vs. segment order-sensitive methods**
  - ⋆ if *high* level of segment overlap between strings, small probability of those segments having occurred in different order

- ⋆ if *low* level of segment overlap between strings, string filtered off by translation utility thresholds

  $\longrightarrow$ very few cases where segment order sensitivity is needed

Additional Results

- Senseval-2 Japanese–English translation memory task
- Character-based indexing with token intersection equivalent to or superior to word-based indexing with segment order-sensitive methods
- The greater the number of translation records, the higher the accuracy


Accuracy (%)
100
90
80
70
60
50
40
30
20
Annotator agreement
AnonymX(MT)
AnonymY1(MT)
CRL-NYU
Ibaraki
Lexical
AnonymY2(TM)
ATR
Kyoto
Structural
Baseline

Overall Conclusions

- Character-based indexing superior to word-based indexing
- Bag-of-words roughly equivalent to segment order-sensitive methods in terms of accuracy *but much faster*, for the string comparison methods considered (naive but fast is good enough)
- N-gram models of local segment contiguity enhance retrieval performance (both speed and accuracy for char-

based indexing)

- The suggested relative running times and accuracies are scalable over variable data sizes

PART III: Enhancements and Extensions to the Basic Paradigm

String Match Stratification

- In traditional TR, string similarity is gauged purely through the lexical make-up of the strings (segmented/stemmed appropriately)

- Possible to add extra strata of information (e.g. stem, POS, semantic classes) and stratify the match process

Dynamic Translation Partitioning

- Traditionally, "translation partitioning" has been according to sentence/logical boundaries
- BUT must allow for case of multiple L1 sentences mapping onto one L2 sentence (and vice versa)
- The coarser the partition granularity, the greater the effect of data sparseness but the lesser the effects of boundary friction
- Possible to fuse together translation fragments for the

portions of translation records best matching discrete portions of the input (cf. EBMT)

Translation Templates

- Traditionally, translation retrieval has been fully lexicalised
- Possibility of generating templates from translation records, e.g. based on pre-defined word classes (e.g. `PART_NAME`, `DATE`)
- Combine "term spotting" with cross-lingually indexed translation templates
- Partition up and cascade translation memories according

to hierarchical word classes (feeding filler translations back into the translation template)

Consistency Checking via Term Alignment

- Align translated document with the source text, and analyse patterns of term translation throughout the document
- Further compare term translation patterns in current document to other documents in the same domain
- Provide feedback to translator on the level of relative translation consistency

Relevance Feedback

- Out of the system outputs, the user will select the translation of highest utility/greatest "recyclability"
- Assuming multiple translation candidates, this provides a form of relevance feedback by way of which segment weights, e.g., can be dynamically reevaluated
- Also possible to dynamically adjust translation utility threshold, or adjust the segment weights to bring translation candidates down under the fixed threshold

Optimising the Translation Process

- Translation is generally carried out serially, based on the original sequence of strings in the source text
- It is possible that a single string towards the end of the text will provide the best translation candidate for multiple strings earlier in the text
- $\longrightarrow$ Treat text as "bag of partitions" and for each input, identify the most similar string out of both the translation memory and source text

- Translate those strings which have the most similar strings in the source text first, and use the resultant translations in translating those similar strings

Dynamic TM Updates via Sentence Alignment

- Traditionally, TM updates have been performed as each input is translated, feeding the translation for the input back into the TM

- In reality, translators make multiple passes over the text incrementally improving readability/translation accuracy as they go

- $\longrightarrow$ Maintain *in situ* update mechanism, but also incrementally update the translation in the TM by performing sentence (partition) alignment between the source text and translation