Data-Intensive Computing with MapReduce

Session 6: Similar Item Detection

Jimmy Lin
University of Maryland
Thursday, February 28, 2013

Developing “big data” intuitions…

Source: Wikipedia (Mahout)

Today's Agenda

- The problem of similar item detection
- Distances and representations
- Minhash
- Random projections
- End-to-end Wikipedia application

What's the Problem?

- Finding similar items with respect to some distance metric
- Two variants of the problem:
  - Offline: extract all similar pairs of objects from a large collection
  - Online: is this object similar to something I've seen before?
- Similarities/differences with the clustering problem

Applications

- Near-duplicate detection of webpages
  - Offline vs. online variant of problem
- Recommender systems
- Forensic text analysis (e.g., plagiarism detection)

Near-Duplicate Detection of Webpages

- What's the source of the problem?
  - Mirror pages (legit)
  - Spam farms (non-legit)
  - Additional complications (e.g., nav bars)
- Naïve algorithm:
  - Compute cryptographic hash for webpage (e.g., MD5)
  - Insert hash values into a big hash table
  - Compute hash for new webpage: collision implies duplicate
- What's the issue?
- Intuition:
  - Hash function needs to be tolerant of minor differences
  - High similarity implies higher probability of hash collision

Literature Note

- Many communities have tackled similar problems:
  - Theoretical computer science
  - Information retrieval
  - Data mining
  - Databases
  - …
- Issues
  - Slightly different terminology
  - Results not easy to compare

Four Steps

- Specify distance metric
  - Jaccard, Euclidean, cosine, etc.
- Compute representation
  - Shingling, tf.idf, etc.
- "Project"
  - Minhash, random projections, etc.
- Extract
  - Bucketing, sliding windows, etc.

Distances

Source: www.flickr.com/photos/thiagoalmeida/250190676/

Distance Metrics

1. Non-negativity:

$$d(x, y) \geq 0$$

2. Identity:

$$d(x, y) = 0 \iff x = y$$

3. Symmetry:

$$d(x, y) = d(y, x)$$

4. Triangle Inequality

$$d(x, y) \leq d(x, z) + d(z, y)$$

Distance: Jaccard

- Given two sets A, B
- Jaccard similarity:

$$\mathrm{J}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$\mathrm{d}(A,B) = 1 - \mathrm{J}(A,B)$$

Distance: Norms

- Given: $\mathrm{x} = [x_1, x_2, \ldots x_n]$
  $\mathrm{y} = [y_1, y_2, \ldots y_n]$

- Euclidean distance (L_2-norm)

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}$$

- Manhattan distance (L_1-norm)

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \sum_{i=0}^{n} |x_i - y_i|$$

- L_r-norm

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \left[\sum_{i=0}^{n} |x_i - y_i|^r \right]^{1/r}$$

Distance: Cosine

- Given: $\mathrm{x} = [x_1, x_2, \ldots x_n]$
  $\mathrm{y} = [y_1, y_2, \ldots y_n]$

- Idea: measure distance between the vectors

$$\cos\theta = \frac{\mathrm{x} \cdot \mathrm{y}}{|\mathrm{x}||\mathrm{y}|}$$

- Thus:

$$\mathrm{sim}(\mathrm{x}, \mathrm{y}) = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2}\sqrt{\sum_{i=0}^{n} y_i^2}}$$

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = 1 - \mathrm{sim}(\mathrm{x}, \mathrm{y})$$

Distance: Hamming

- Given two bit vectors
- Hamming distance: number of elements which differ

Representations

Representations: Text

- Unigrams (i.e., words)
- Shingles = *n*-grams
  - At the word level
  - At the character level
- Feature weights
  - boolean
  - tf.idf
  - BM25
  - …
- Which representation to use?
- What's the actual text?

Use the hashing trick!

Representations: Beyond Text

- For recommender systems:
  - Items as features for users
  - Users as features for items
- For graphs:
  - Adjacency lists as features for vertices
- With log data:
  - Behaviors (clicks) as features

Minhash

Source: www.flickr.com/photos/rheinitz/6158837748/

Minhash

- Seminal algorithm for near-duplicate detection of webpages
  - Used by AltaVista
  - For details see Broder et al. (1997)
- Setup:
  - Documents (HTML pages) represented by shingles (*n*-grams)
  - Jaccard similarity: dups are pairs with high similarity

Preliminaries: Representation

- Sets:
  - A = $\{e_1, e_3, e_7\}$
  - B = $\{e_3, e_5, e_7\}$
- Can be equivalently expressed as matrices:

Element	A	B
e_1	1	0
e_2	0	0
e_3	1	1
e_4	0	0
e_5	0	1
e_6	0	0
e_7	1	1

Preliminaries: Jaccard

Element	A	B
e_1	1	0
e_2	0	0
e_3	1	1
e_4	0	0
e_5	0	1
e_6	0	0
e_7	1	1

Let:

M_{00} = # rows where both elements are 0

M_{11} = # rows where both elements are 1

M_{01} = # rows where A=0, B=1

M_{10} = # rows where A=1, B=0

$$\mathrm{J}(A,B) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Minhash

- Computing minhash
  - Start with the matrix representation of the set
  - Randomly permute the rows of the matrix
  - minhash is the first row with a "one"
- Example:

$h(A) = e_3$ $h(B) = e_5$

Element	A	B
e_1	1	0
e_2	0	0
e_3	1	1
e_4	0	0
e_5	0	1
e_6	0	0
e_7	1	1

Element	A	B
e_6	0	0
e_2	0	0
e_5	0	1
e_3	1	1
e_7	1	1
e_4	0	0
e_1	1	0

Minhash and Jaccard

Element	A	B	
e_6	0	0	M_{00}
e_2	0	0	M_{00}
e_5	0	1	M_{01}
e_3	1	1	M_{11}
e_7	1	1	M_{11}
e_4	0	0	M_{00}
e_1	1	0	M_{10}

$$P[h(A) = h(B)] = \mathrm{J}(A, B)$$

$$\frac{M_{11}}{M_{01} + M_{10} + M_{11}} \qquad \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

To Permute or Not to Permute?

- Permutations are expensive
- Interpret the hash value as the permutation
- Only need to keep track of the minimum hash value
  - Can keep track of multiple minhash values at once

Extracting Similar Pairs (LSH)

- We know: $P[h(A) = h(B)] = \mathrm{J}(A, B)$
- Task: discover all pairs with similarity greater than *s*
- Algorithm:
  - For each object, compute its minhash value
  - Group objects by their hash values
  - Output all pairs within each group
- Analysis:
  - Probability we will discovered all pairs is *s*
  - Probability that any pair is invalid is (1 – s)
- What's the fundamental issue?

Two Minhash Signatures

- Task: discover all pairs with similarity greater than s
- Algorithm:
  - For each object, compute two minhash values and concatenate together into a signature
  - Group objects by their signatures
  - Output all pairs within each group
- Analysis:
  - Probability we will discovered all pairs is s^2
  - Probability that any pair is invalid is $(1 - s)^2$

k Minhash Signatures

- Task: discover all pairs with similarity greater than s
- Algorithm:
  - For each object, compute k minhash values and concatenate together into a signature
  - Group objects by their signatures
  - Output all pairs within each group
- Analysis:
  - Probability we will discovered all pairs is s^k
  - Probability that any pair is invalid is $(1 - s)^k$
- What's the issue now?

n different *k* Minhash Signatures

- Task: discover all pairs with similarity greater than s
- Algorithm:
  - For each object, compute n sets k minhash values
  - For each set, concatenate k minhash values together
  - Within each set:
    - Group objects by their signatures
    - Output all pairs within each group
  - De-dup pairs
- Analysis:
  - Probability we will miss a pair is $(1 - s^k)^n$
  - Probability that any pair is invalid is $n(1 - s)^k$

Practical Notes

- In some cases, checking all candidate pairs may be possible
  - Time cost is small relative to everything else
  - Easy method to discard false positives
- Most common practical implementation:
  - Generate *M* minhash values, randomly select *k* of them *n* times
  - Reduces amount of hash computations needed
- Determining "authoritative" version is non-trivial

MapReduce Implementation

- Map over objects:
  - Generate *M* minhash values, randomly select *k* of them *n* times
  - Each draw yields a signature: emit as intermediate key, value is object id
- Shuffle/sort:
- Reduce:
  - Receive all object ids with same signature, generate candidate pairs
  - Okay to buffer in memory?
- Second pass to de-dup

Offline Extraction vs. Online Querying

- Batch formulation of the problem:
  - Discover all pairs with similarity greater than s
  - Useful for post-hoc batch processing of web crawl
- Online formulation of the problem:
  - Given new webpage, is it similar to one I've seen before?
  - Useful for incremental web crawl processing

Online Similarity Querying

- Preparing the existing collection:
  - For each object, compute *n* sets of *k* minhash values
  - For each set, concatenate *k* minhash values together
  - Keep each signature in hash table (in memory)
  - Note: can parallelize across multiple machines
- Querying and updating:
  - For new webpage, compute signatures and check for collisions
  - Collisions imply duplicate (determine which version to keep)
  - Update hash tables

Source: www.flickr.com/photos/roj/4179478228/

Limitations of Minhash

- Minhash is great for near-duplicate detection
  - Set high threshold for Jaccard similarity
- Limitations:
  - Jaccard similarity only
  - Set-based representation, no way to assign weights to features
- Random projections:
  - Works with arbitrary vectors using cosine similarity
  - Same basic idea, but details differ
  - Slower but more accurate: no free lunch!

Random Projection Hashing

- Generate a random vector *r* of unit length
  - Draw from univariate Gaussian for each component
  - Normalize length
- Define:

$$h_r(\mathrm{u}) = \begin{cases} 1 & \text{if } \mathrm{r} \cdot \mathrm{u} \geq 0 \\ 0 & \text{if } \mathrm{r} \cdot \mathrm{u} < 0 \end{cases}$$

- Physical intuition?

RP Hash Collisions

- It can be shown that:

$$P[h_r(\mathrm{u}) = h_r(\mathrm{v})] = 1 - \frac{\theta(\mathrm{u}, \mathrm{v})}{\pi}$$

  - Proof in (Goemans and Williamson, 1995)
- Physical intuition?

Random Projection Signature

- Given *D* random vectors:

$$[\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3, \ldots \mathrm{r}_D]$$

- Convert each object into a *D* bit signature

$$\mathrm{u} \rightarrow [h_{r_1}(\mathrm{u}), h_{r_2}(\mathrm{u}), h_{r_3}(\mathrm{u}), \ldots h_{r_D}(\mathrm{u})]$$

  - We can derive:

$$\mathrm{sim}(\mathrm{u}, \mathrm{v}) = \cos\left[\left(\frac{\pi}{D}\right) \mathrm{hamming}(\mathrm{s_u}, \mathrm{s_v})\right]$$

- Thus: similarity boils down to comparison of hamming distances between signatures

One-RP Signature

- Task: discover all pairs with cosine similarity greater than s
- Algorithm:
  - Compute D-bit RP signature for every object
  - Take first bit, bucket objects into two sets
  - Perform brute force pairwise (hamming distance) comparison in each bucket, retain those below hamming distance threshold
- Analysis:
  - Probability we will discover all pairs:*

$$1 - \frac{\cos^{-1}(s)}{\pi}$$

  - Efficiency:

$$N^2 \qquad \text{vs.} \qquad 2\left(\frac{N}{2}\right)^2$$

* Note, this is actually a simplification: see Ture et al. (SIGIR 2011) for details.

Two-RP Signature

- Task: discover all pairs with cosine similarity greater than s
- Algorithm:
  - Compute D-bit RP signature for every object
  - Take first two bits, bucket objects into four sets
  - Perform brute force pairwise (hamming distance) comparison in each bucket, retain those below hamming distance threshold
- Analysis:
  - Probability we will discover all pairs:

$$\left[1 - \frac{\cos^{-1}(s)}{\pi}\right]^2$$

  - Efficiency:

$$N^2 \qquad \text{vs.} \qquad 4\left(\frac{N}{4}\right)^2$$

k-RP Signature

- Task: discover all pairs with cosine similarity greater than *s*
- Algorithm:
  - Compute *D*-bit RP signature for every object
  - Take first *k* bits, bucket objects into 2^k sets
  - Perform brute force pairwise (hamming distance) comparison in each bucket, retain those below hamming distance threshold
- Analysis:
  - Probability we will discover all pairs:

$$\left[1 - \frac{\cos^{-1}(s)}{\pi}\right]^k$$

  - Efficiency:

$$N^2 \qquad \text{vs.} \qquad 2^k \left(\frac{N}{2^k}\right)^2$$

m Sets of *k*-RP Signature

- Task: discover all pairs with cosine similarity greater than *s*
- Algorithm:
  - Compute *D*-bit RP signature for every object
  - Choose *m* sets of *k* bits
  - For each set, use *k* selected bits to partition objects into 2^k sets
  - Perform brute force pairwise (hamming distance) comparison in each bucket (of each set), retain those below hamming distance threshold
- Analysis:
  - Probability we will discover all pairs:

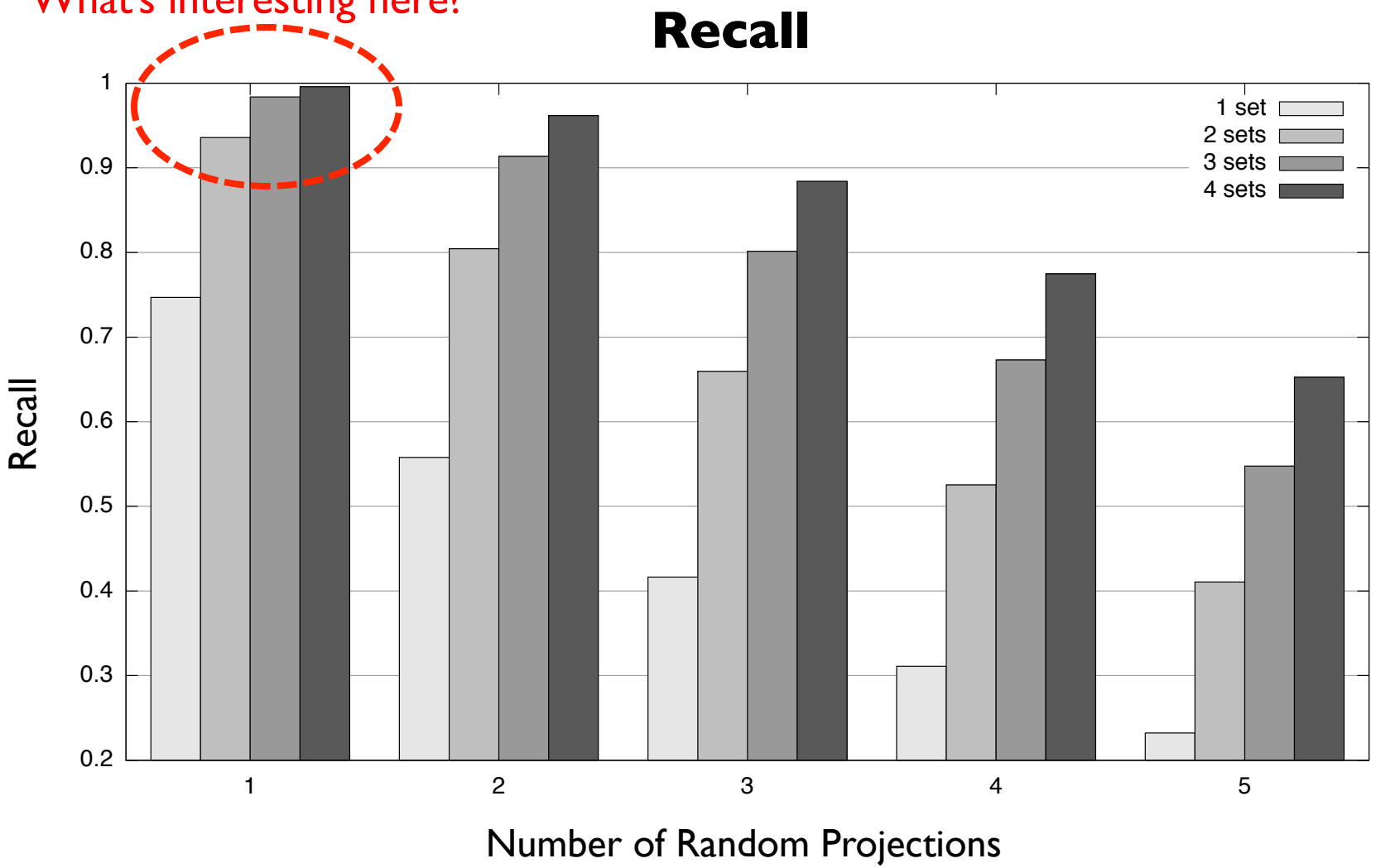
$$1 - \left[1 - \left[1 - \frac{\cos^{-1}(s)}{\pi}\right]^k\right]^m$$

  - Efficiency: $N^2 \quad \text{vs.} \quad m \cdot 2^k \left(\frac{N}{2^k}\right)^2$

Theoretical Results

Threshold = 0.7

What's interesting here?



Online Querying

- Preprocessing:
  - Compute *D*-bit RP signature for every object
  - Choose *m* sets of *k* bits and use to bucket
  - Store signatures in memory (across multiple machines)
- Querying
  - Compute *D*-bit signature of query object, choose *m* sets of *k* bits in same way
  - Perform brute-force scan of correct bucket (in parallel)

Additional Issues to Consider

- Emphasis on recall, not precision
- Two sources of error:
  - From LSH
  - From using hamming distance as proxy for cosine similarity
- Balance of bucket sizes?
- Parameter tuning?

“Sliding Window” Algorithm

- Compute *D*-bit RP signature for every object
- For each object, permute bit signature *m* times
- For each permutation, sort bit signatures
  - Apply sliding window of width *B* over sorted
  - Compute hamming distances of bit signatures within window

MapReduce Implementation

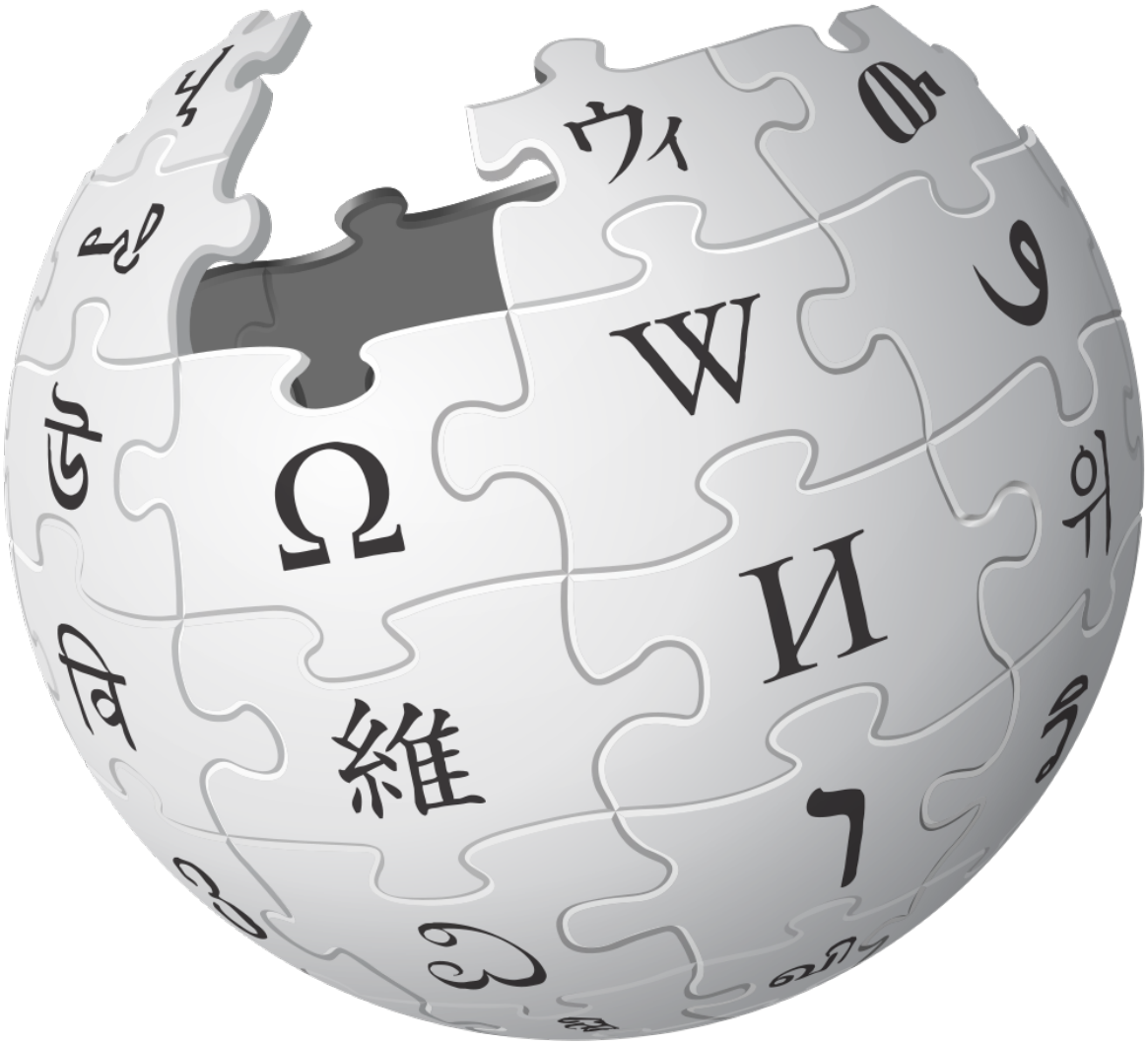
- Mapper:
  - Process each individual object in parallel
  - Load in random vectors as side data
  - Compute bit signature
  - Permute *m* times, for each emit:
    key = (*n*, signature), where *n* = [1 … *m*]
    value = object id
- Reduce
  - Keep FIFO queue of *B* bit signatures
  - For each newly-encountered bit signature, compute hamming distance wrt all bit signatures in queue
  - Add new bit signature to end of queue, displacing oldest

Slightly Faster MapReduce Variant

- In reducer, don't perform hamming distance computations
  - Write directly to HDFS in large blocks
- Perform hamming distance computations in second mapper-only pass
  - Advantages?

Case Study: Wikipedia

Ture et al., SIGIR 2011
Ture and Lin, NAACL 2012

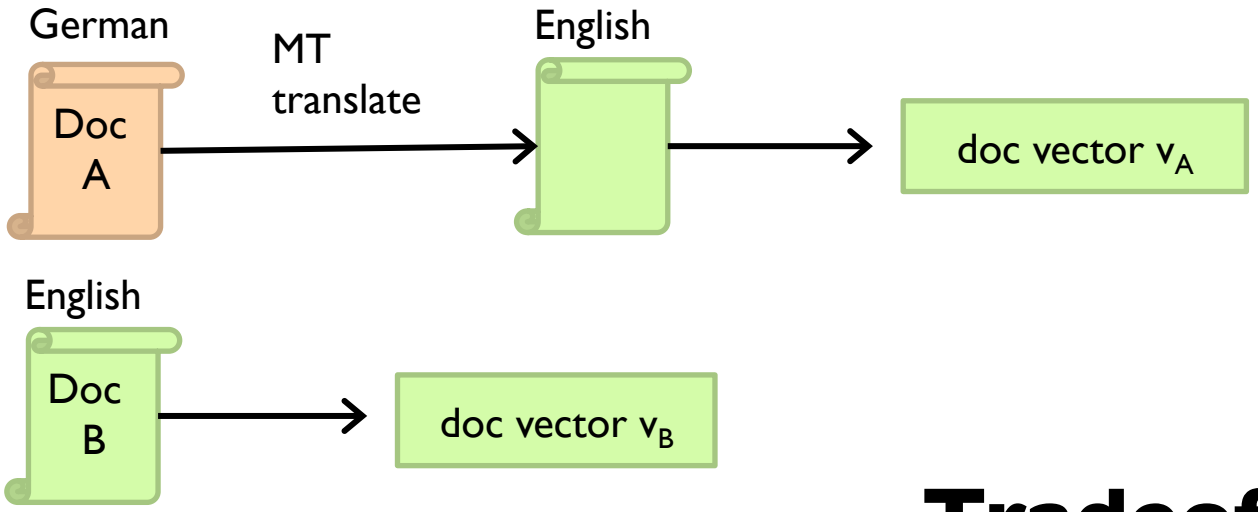


Source: Wikipedia (Wikipedia)

Mining Bitext from Wikipedia

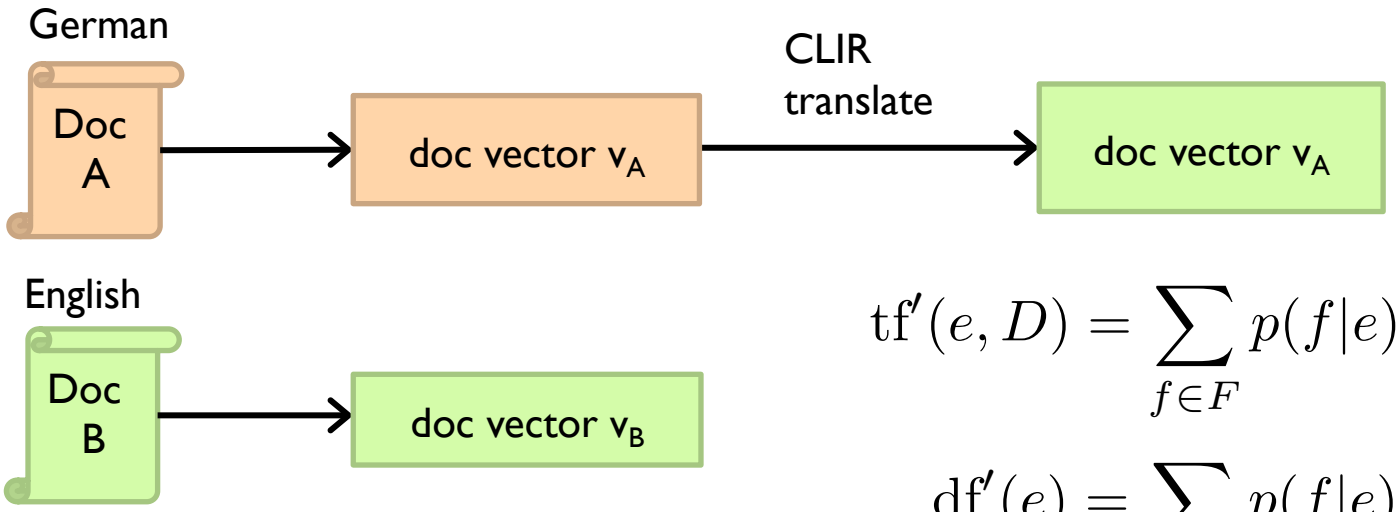
- Task: extract similar article pairs in *different* languages
  - Why is this useful?
  - Why not just use inter-wiki links?
- Use LSH!
  - Why is this non-trivial?

Machine Translation



Tradeoffs?

"Vector" Translation

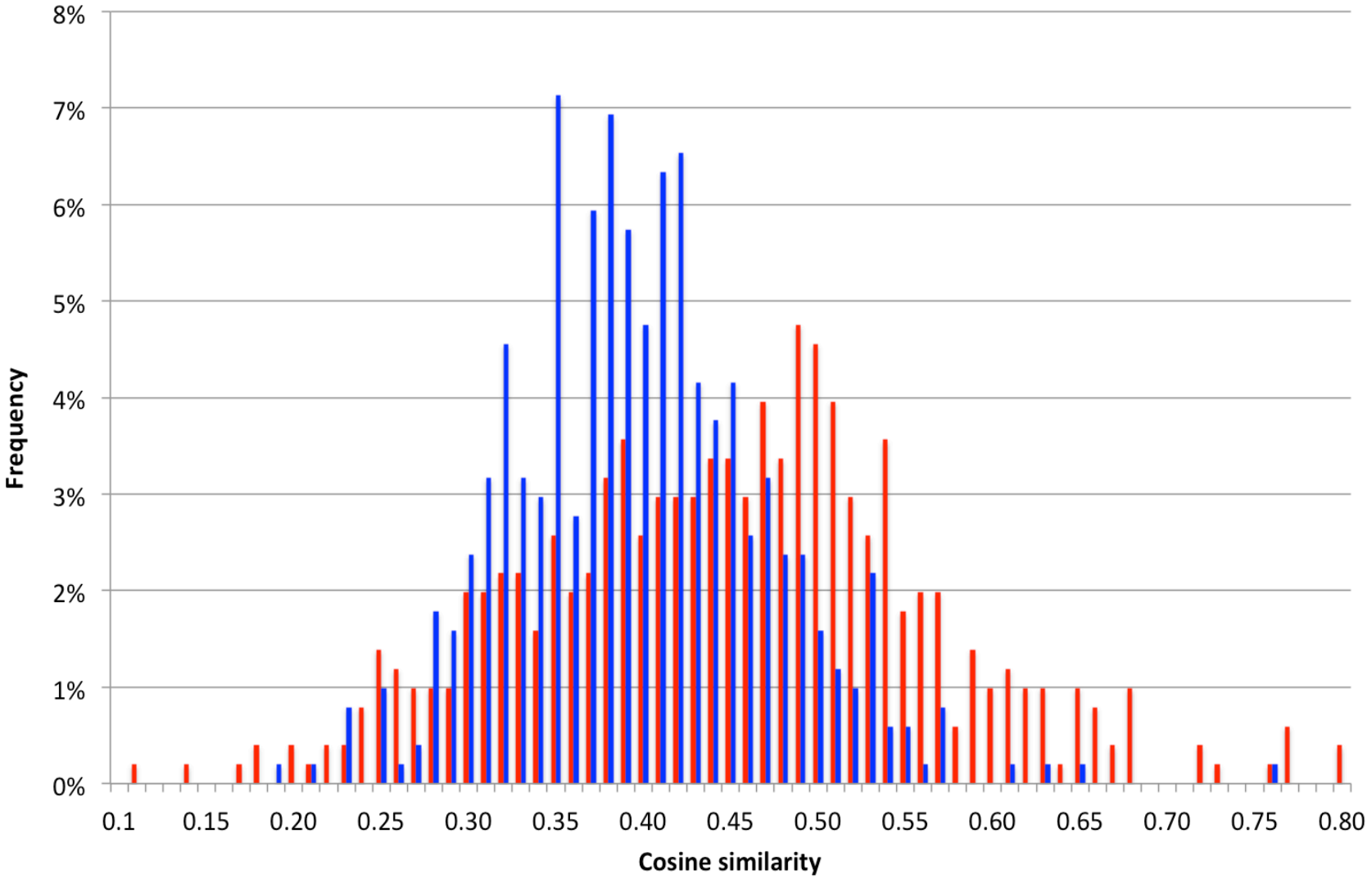


$$\mathrm{tf}'(e, D) = \sum_{f \in F} p(f|e) \cdot \mathrm{tf}(f, D)$$

$$\mathrm{df}'(e) = \sum_{f \in F} p(f|e) \cdot \mathrm{df}(f)$$

Translations are noisy!

Question: To what extent are "ground truth" mutual translation pairs similar?



Architecture

German articles

English articles

Preprocess

document vectors (English)

Signature generation

Signatures

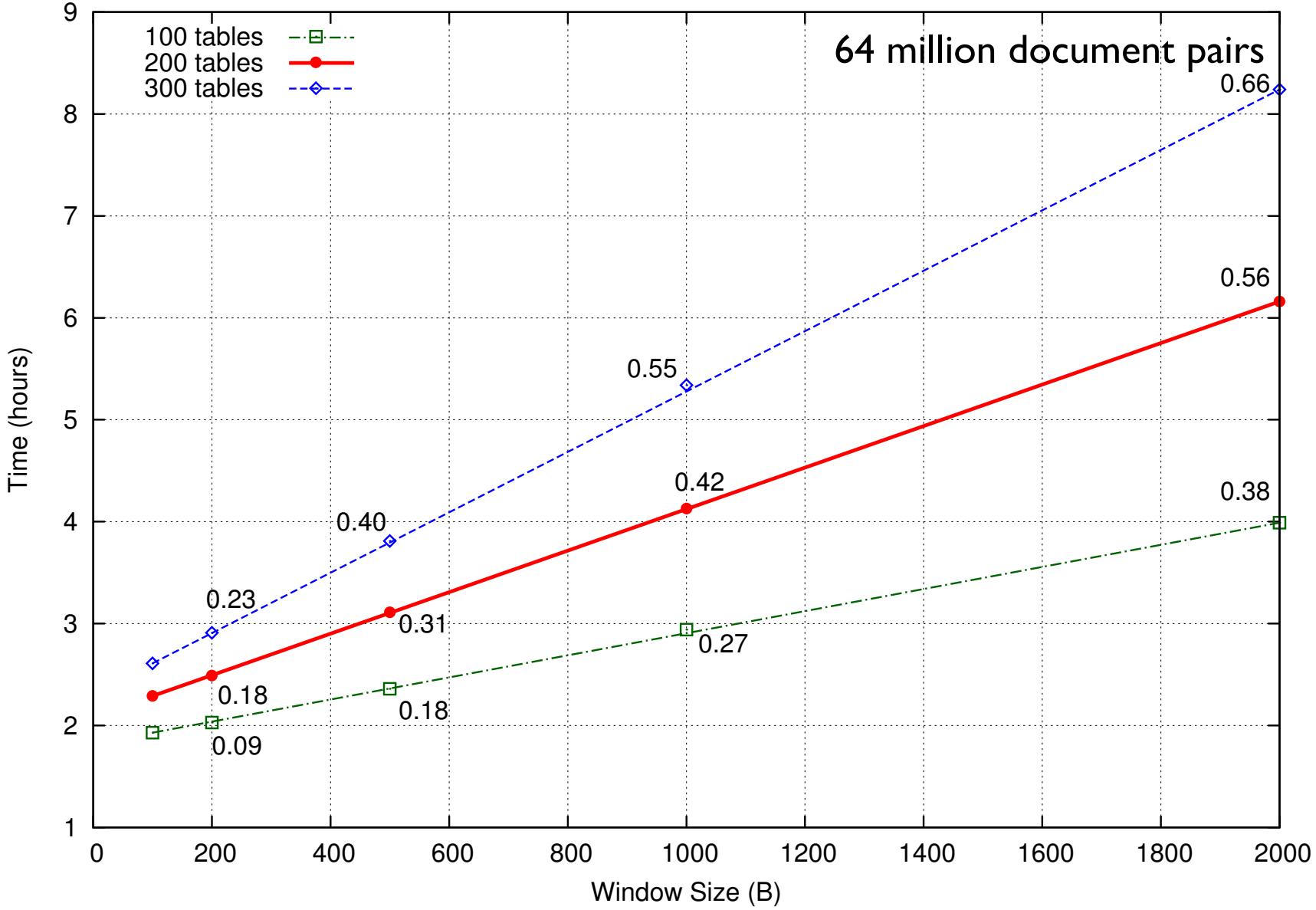
LSH

Similar article pairs

Evaluation Setup

- Collection: 3.44m English + 1.47m German Wikipedia
  - Brute force: 5.05 trillion comparisons
- Task: extract similar documents with cosine similarity > 0.3
- Representation: 1000 bit random projections
- Ground truth:
  - Sample ~1000 German Wikipedia articles
- Evaluation metrics:
  - Time
  - Relative cost
  - Recall

Results (16 node cluster)

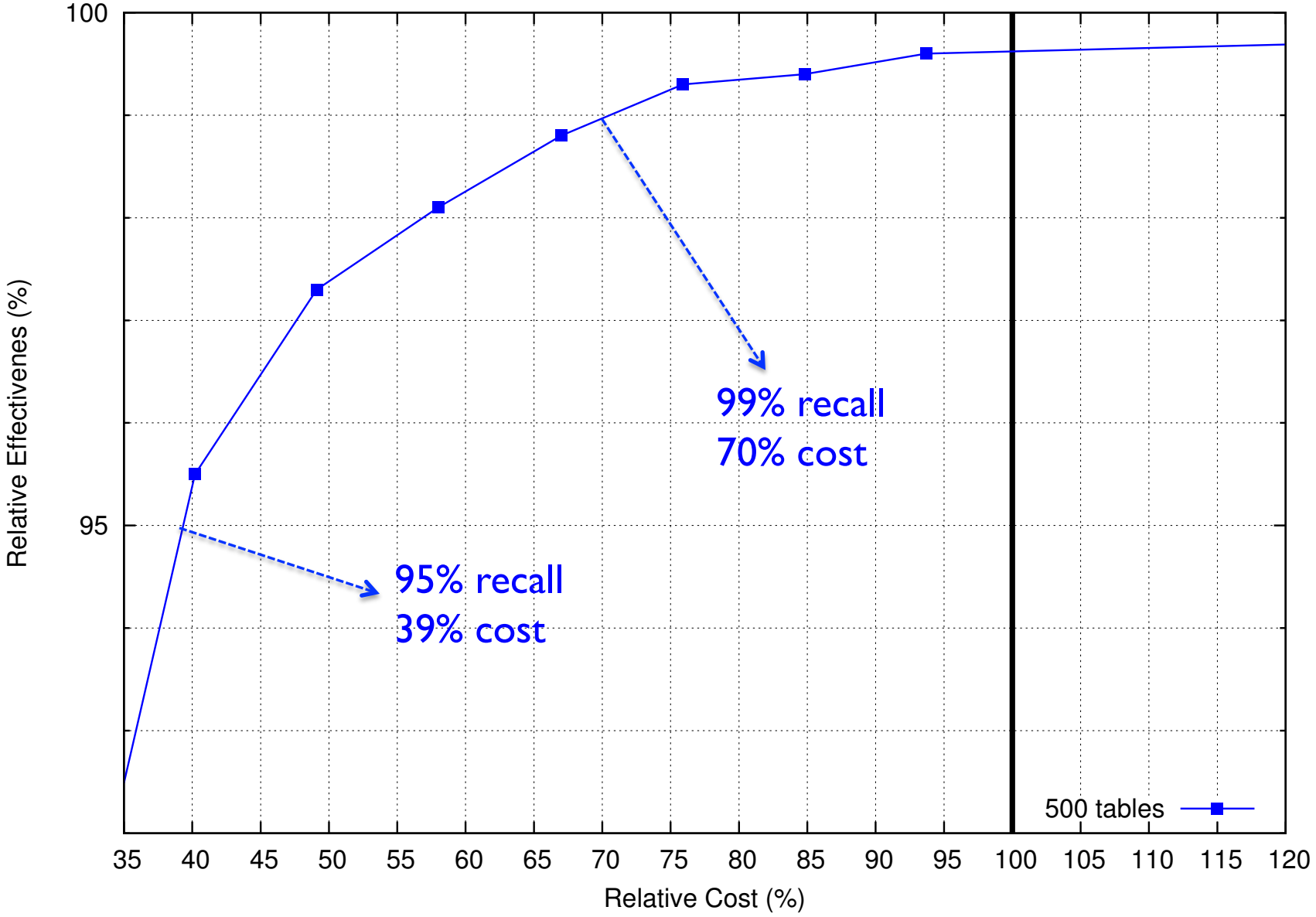


Sources of Error

- Hamming distance computation introduces error!
  - 1000-bit random projections yields ~0.04 absolute error
  - Maximum obtainable recall is 0.76
- Why not just compute cosine similarity?
  - 20 times slower than computing hamming distance

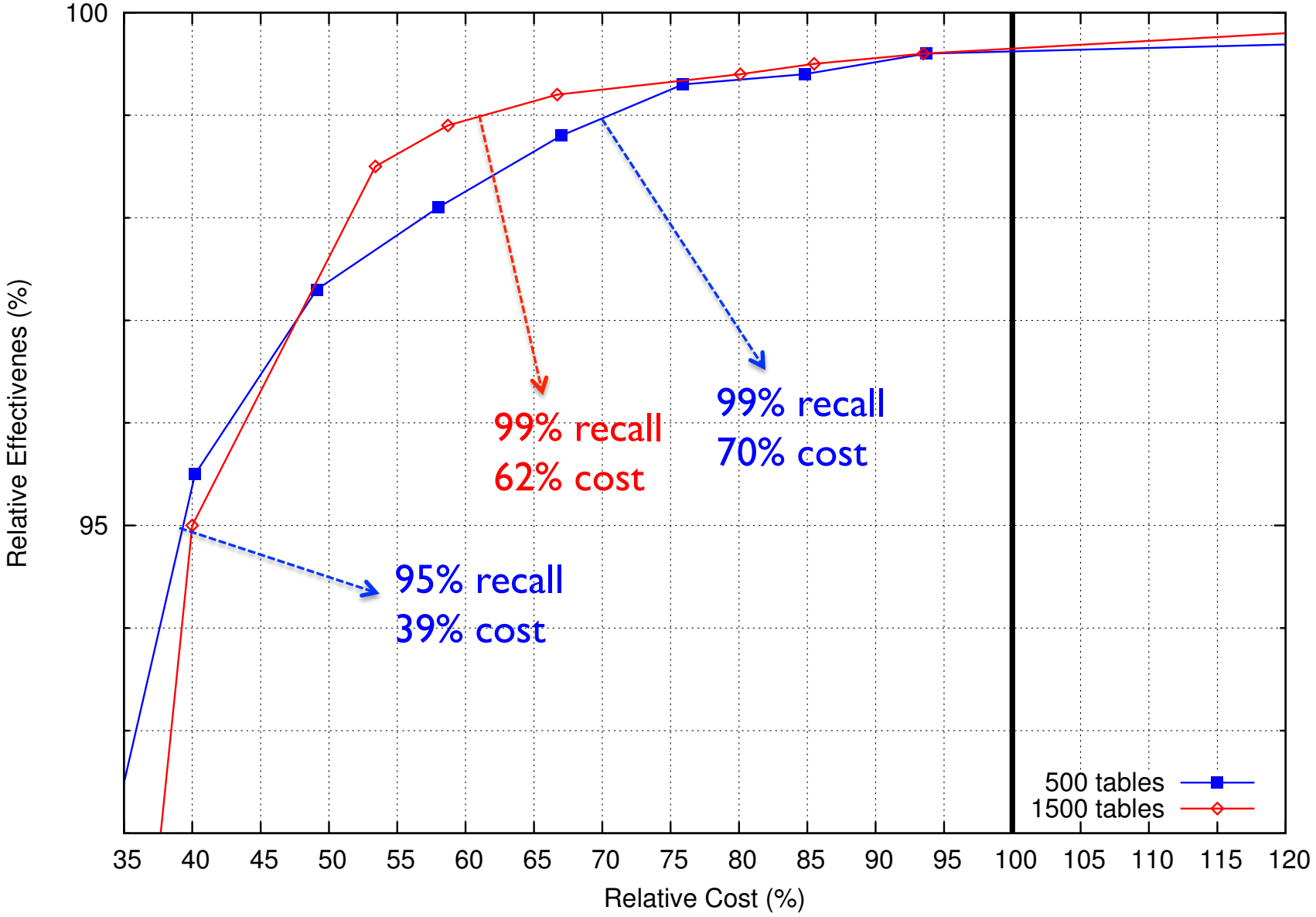
Let's renormalize with respect to hamming distance upper bound…

No Free Lunch!



No Free Lunch!

100% recall? Don't use LSH!



Parallel Sentence Extraction

- Model parallel sentence extraction as a classification problem
  - Given a candidate pair, classify as "parallel" or "not parallel"
- What's the problem?
  - 64m document pairs yields 400b raw sentence pairs

What's that quote about premature optimization?

Two-Step Classification

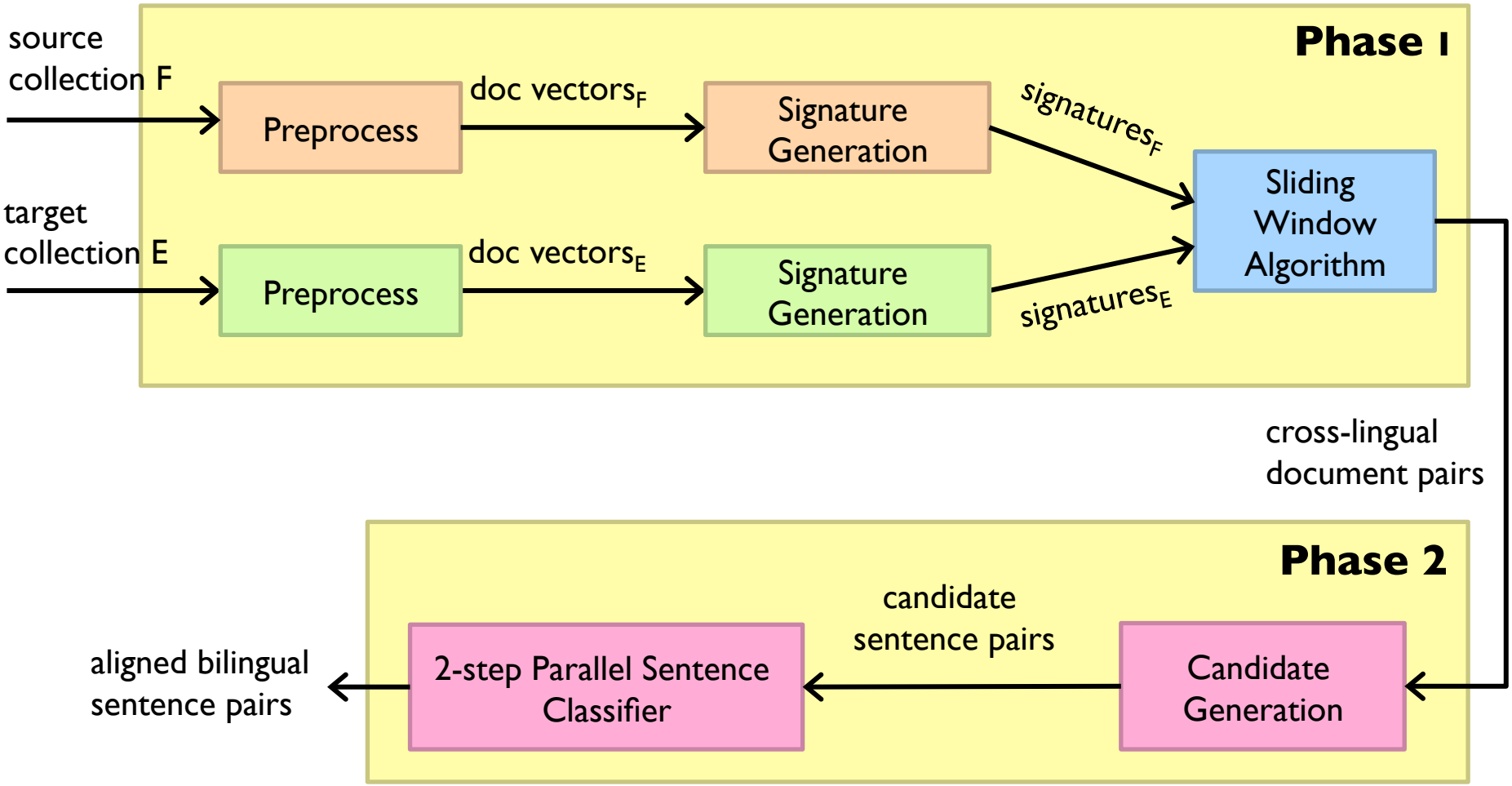
- Approach:
  - Apply a few simple heuristics
  - "Simple" classifier: efficiently filters out irrelevant pairs
  - "Complex" classifier: effectively classifies remaining pairs

Bitext Classifier	**Measure**	**In-domain** (Europarl)	**Out-of-domain** (Wikipedia)	**Efficiency** (per instance, in μs)
Simple	Recall @ P95	0.81	0.57	27
	Recall @ P80	0.90	0.76	
	Best F-score	0.88	0.78	
Complex	Recall @ P95	0.87	0.55	105
	Recall @ P80	0.94	0.80	
	Best F-score	0.91	0.81	

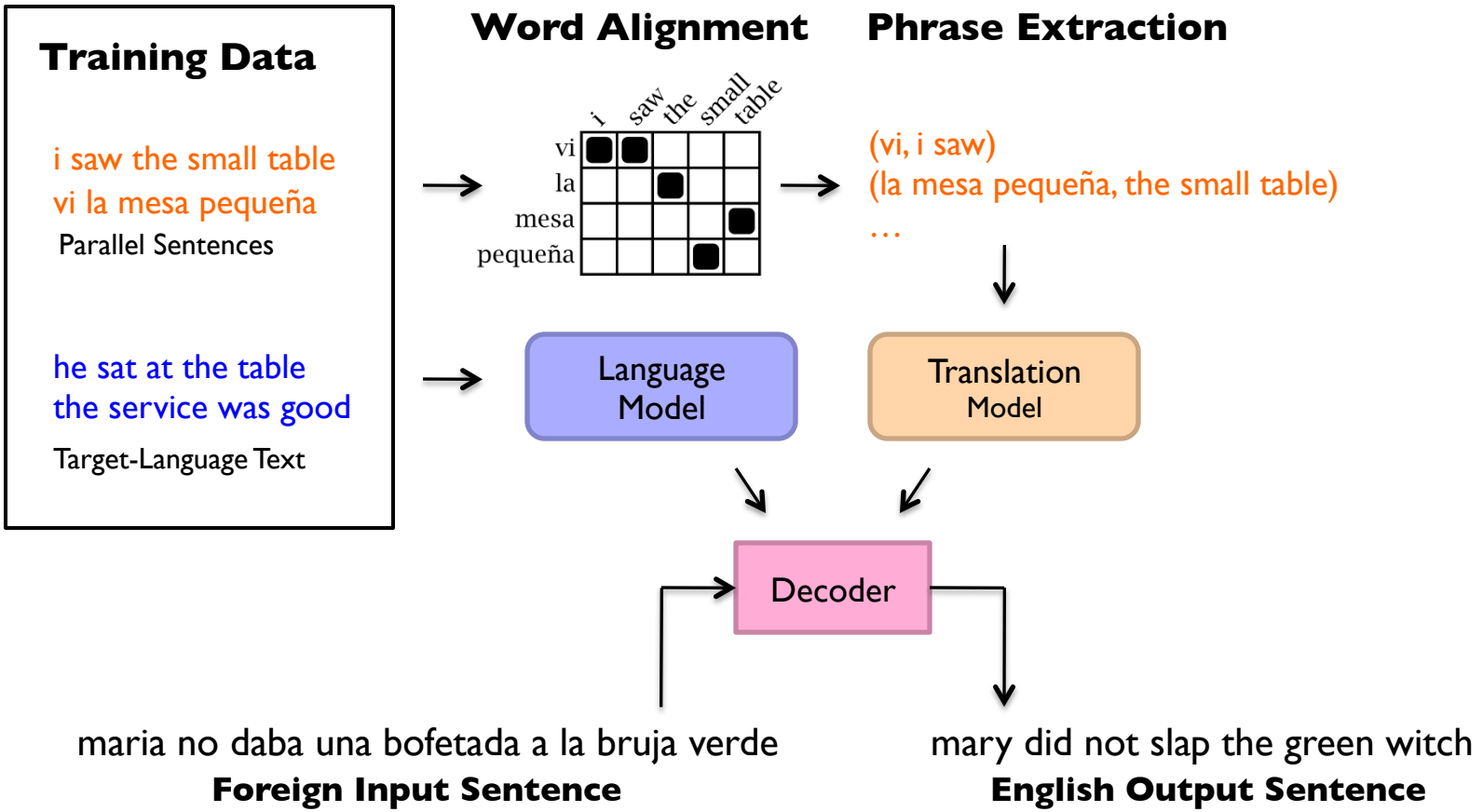
Join

- Algorithm 1
  - Mapper processes both English and foreign collection
    - Load all (d_e, d_f) similarity pairs in memory as side data
    - When encountering relevant document, emit (d_e, d_f) pair as key, list of sentence vectors as value
  - Reducer:
    - Gather sentence vectors, compute Cartesian product and apply classifier
- Algorithm 2
  - Same idea as first, except with stripes pattern
- Which is faster?

Complete Pipeline

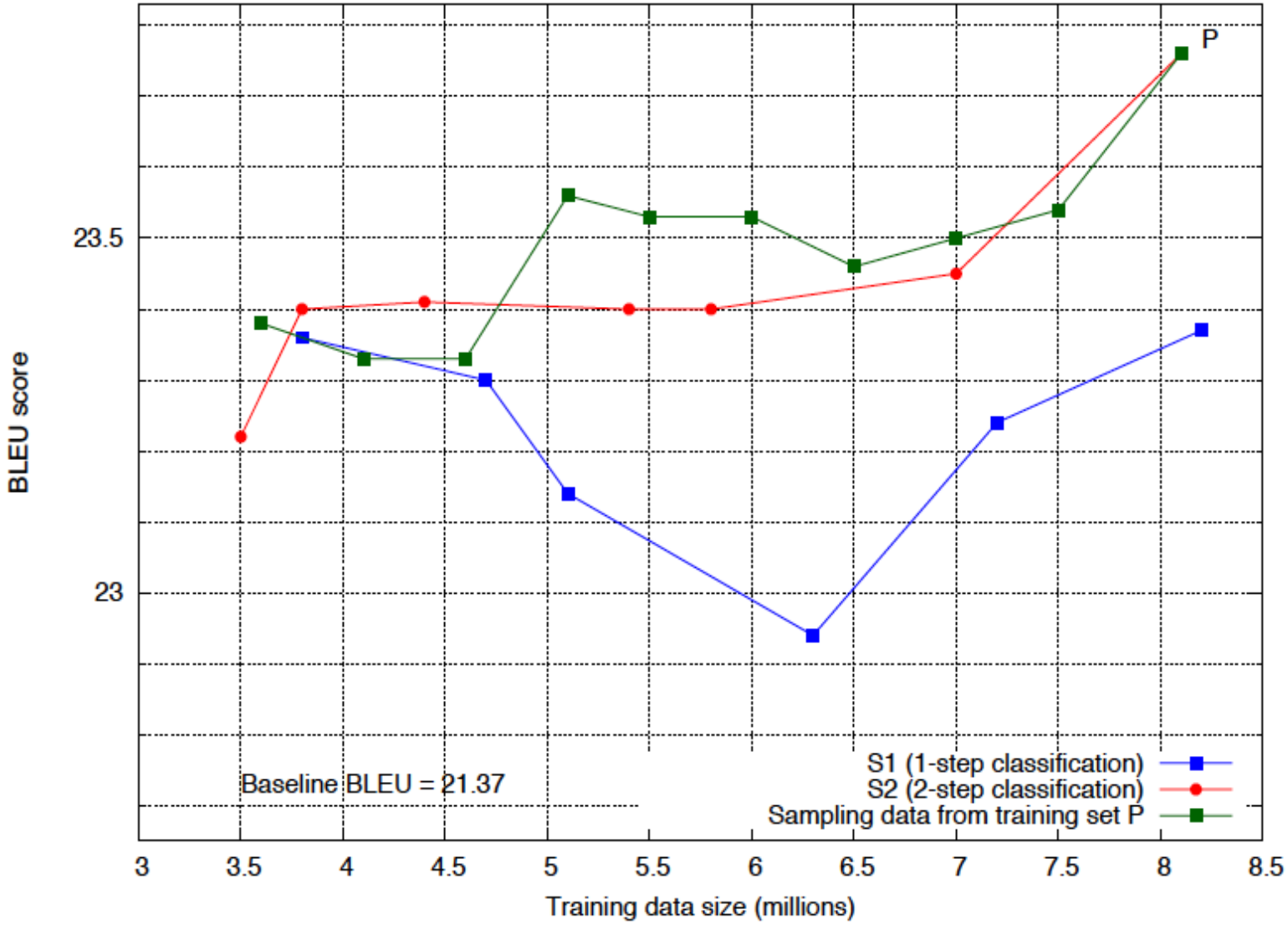


Statistical Machine Translation



$$\hat{e}_1^I = \arg\max_{e_1^I} \left[P(e_1^I \mid f_1^J) \right] = \arg\max_{e_1^I} \left[P(e_1^I) P(f_1^J \mid e_1^I) \right]$$

Evaluation


BLEU score
23.5
23
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
Training data size (millions)
P
Baseline BLEU = 21.37
S1 (1-step classification)
S2 (2-step classification)
Sampling data from training set P

Today's Agenda

- The problem of similar item detection
- Distances and representations
- Minhash
- Random projections
- End-to-end Wikipedia application

Source: Wikipedia (Japanese rock garden)