“ALEXANDRU IOAN CUZA” UNIVERSITATY OF IAŞI
FACULTY OF COMPUTER SCIENCE

How to add a new language on the NLP map: Tools and resources you can build

Daniela GÎFU

http://profs.info.uaic.ro/~daniela.gifu/

1. Monolingual NLP. Building resources and tools for a new language

- Automatic construction of a corpus
- Construction of basic resources and tools starting with a corpus:

- language models
- unsupervised syntactic analysis – POS tagging
- clustering of similar entities – words, phrases, texts

- Applications:

- spelling correction – diacritics restoration
- language identification
- language models for information retrieval and text classification

Applications: Diacritics Restoration

- How important are diacritics?
  - Romanian:

    paturi – pături peste – peşte

    (beds – blankets over/above – fish)

  - Other languages?

Diacritics – in European languages with Latin-based alphabets

- Albanian, Basque, Breton, Catalan, Czech, Danish, Dutch, Estonian, Faroese, Finnish, French, Gaelic, German, Hungarian, Icelandic, Italian, Lower Sorbian, Maltese, Norwegian, Polish, Portuguese, Romanian, Sami, Serbo-Croatian, Slovak, Slovene, Spanish, Swedish, Turkish, Upper-Sorbian, Welsh

Diacritics – in European Languages with Latin-based Alphabets

- 31 (at least) European languages have between 2 and 20 diacritics
  - Albanian: ç ë
  - Basque: ñ ü
  - Dutch: á à â ä é è ê ë í ì î ï ó ò ô ö ú ù û ü
- English has diacritics for few words, imported from other languages (fiancé, café, …)
  - has instead a much larger number of homonyms

Restoring Diacritics

- word level
  - **requires**:
    - dictionaries
    - processing tools (part-of-speech taggers, parsers)
    - large corpora from which to learn a language model
  - **obtain rules such as**:
    - “anuncio” should change to “anunció” when it is a verb

Restoring Diacritics

- letter level
  - requires a small corpus in which to observe letter sequences
  - obtain rules such as:
    - *"s" followed by "i" and blank space, and preceded by a blank space should change to "ş"*

- This approach should work well for unknown words, and without requiring tools for morphologic and syntactic analysis

Letter Level

- letters are the smallest level of granularity in language analysis
- instead of dealing with 100 000+ units (words), we have more or less 26 characters to deal with
- language independence!

Restoring Diacritics - Experiments

- Data
  - medium sized corpus with diacritics in the language of choice
- Learning
  - we learn to choose between possible diacritics
  - Tilburg Memory Based Learning:
    (TiMBL) - https://languagemachines.github.io/timbl/
  - features: surrounding letters – 5 letters before and after the diacritic (including space, punctuation, and other characters)

l, i, n, , (, s, u, b, , i, n
g, a, r, d, i, s, t, u, l, ,,
e, , o, r, a, s, ., t, o, t

- Results: EXCELLENT! – average 98% (tested on several languages)

Problematic Diacritics

- diacritics that distinguish between definite/indefinite article inflections
  - Romanian:

    masă - *masa (table – the table)*

- What happens to *peste – peşte*?
  - the number of surrounding letters we look at allows us to go across word boundaries

1. Monolingual NLP. Building resources and tools for a new language

- Automatic construction of a corpus
- Construction of basic resources and tools starting with a corpus:
  - language models
  - unsupervised syntactic analysis – POS tagging
  - clustering of similar entities – words, phrases, texts
- Applications:
  - spelling correction – diacritics restoration
  - **language identification**
  - language models for information retrieval and text classification

Language Identification

- Determine the language of an unknown text

- N-grams models are very effective solutions for this problem
  - *Letter-based models*
  - *Word-based models*
  - *Smoothing*

Training Letter-based Language Models

- Learn a letter bigram/n-gram model, with letter bigram/n-gram probabilities
- A separate language model has to be learned for each language
- Example:
  - Je ne sais pas => P(e|J), … , P(a|s) …
  - I don’t know
  - Nu stiu

Language Identification of a New Text

- Apply each individual model to determine the most likely language for the test text
- I.e. determine the probability associated with each sequence of letters in the test file
- Example:
  - Astazi este luni: P(s|A) * P(t|s) * …
    - use log
  - Find sequence probability for each trained language model
  - Choose the language with the highest probability

Word-based Language Models

- Could also use unigrams
- Requires smoothing
  - Avoid 0 probabilities
- Example
  - Astazi este luni:

    P(astazi) * P(este) * P(luni)
  - Compute probability with respect to each trained word-based language model
  - Find the language that leads to the highest probability

1. Monolingual NLP. Building resources and tools for a new language

- Automatic construction of a corpus
- Construction of basic resources and tools starting with a corpus:
  - language models
  - unsupervised syntactic analysis – POS tagging
  - clustering of similar entities – words, phrases, texts
- Applications:
  - spelling correction – diacritics restoration
  - language identification
  - **language models for information retrieval and text classification**

Information Retrieval

- Given a query Q, and a collection of documents D, find the document Di that is the most “similar” to the query Q
- Example
  - Q: “eurolan summer school”
  - Most similar document: http://www.cs.ubbcluj.ro/eurolan2005/
- IR systems:
    - Google, Yahoo, Altavista

Text Classification

- Given a set of documents classified into C categories, and a new document D, find the category that is most appropriate for the new document
- Example
  - Categories: arts, music, computer science
  - Document: http://www.cs.ubbcluj.ro/eurolan2005/
  - Most appropriate category: computer science

Information Retrieval and Text Classification

- Challenges
  - Find similarities between texts
    - IR: Query and Document
    - TC: Document and Document
  - Weight terms in texts
    - Discount “the” but emphasize “language”
    - Use language models
  - Vector-space model

Vector-Space Model

- *t* distinct terms remain after preprocessing
  - Unique terms that form the VOCABULARY
- These "orthogonal" terms form a vector space

  Dimension = *t* = |vocabulary|
  - 2 terms → bi-dimensional; …; n-terms → n-dimensional
- Each term, *i*, in a document or query *j*, is given a real-valued weight, $w_{ij.}$
- Both documents and queries are expressed as t-dimensional vectors:

  $d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$

Vector-Space Model

Query as vector:

- Regard query as short document
- Return the documents ranked by the closeness of their vectors to the query, also represented as a vector

- Note
  - Vectorial model was developed in the SMART system (Salton, c. 1970)

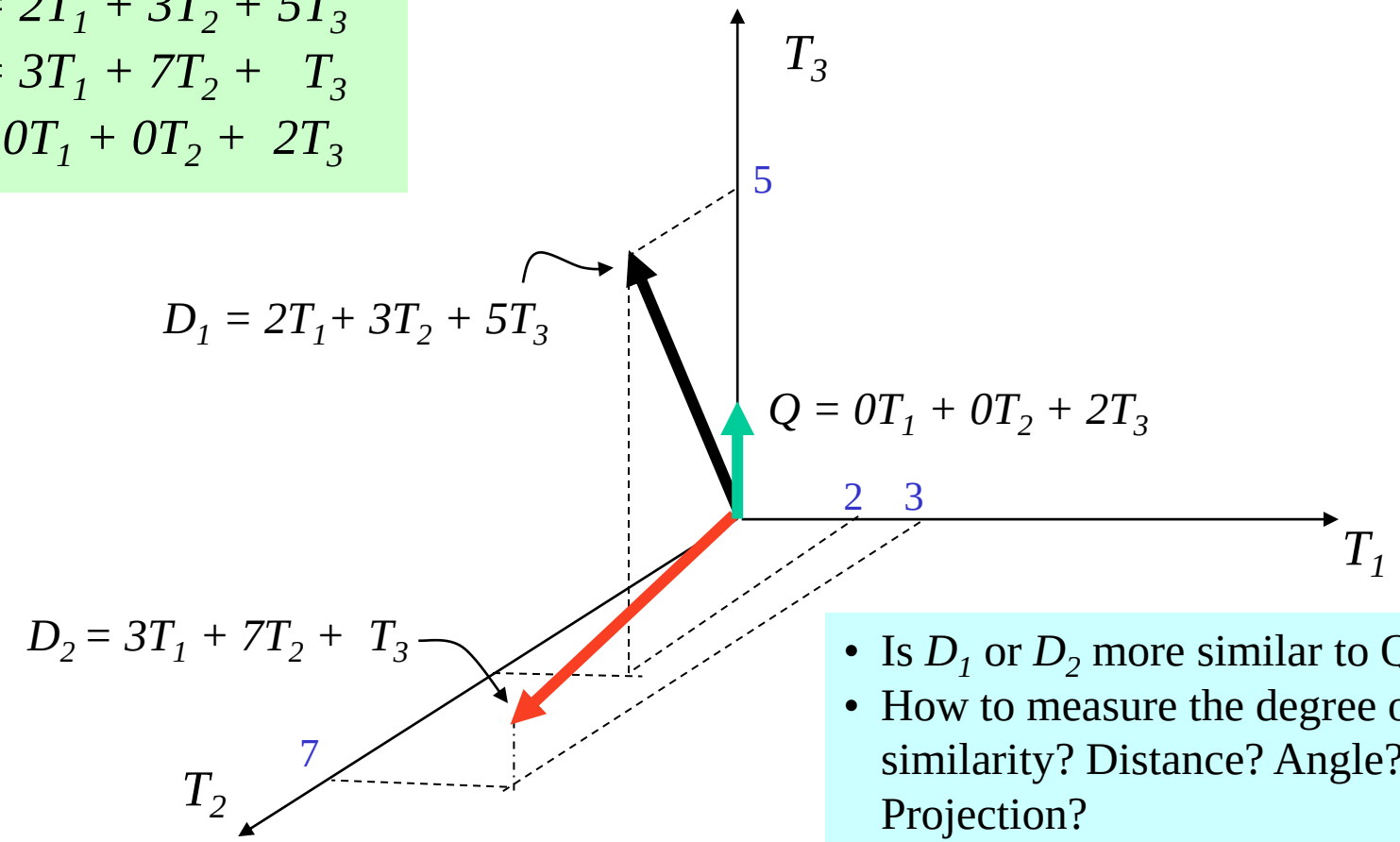
Graphic Representation

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

Document Collection Representation

- A collection of *n* documents can be represented in the vector space model by a term-document matrix.
- An entry in the matrix corresponds to the "weight" of a term in the document; zero means the term has no significance in the document or it simply doesn't exist in the document.

$$\begin{array}{c|cccc} & T_1 & T_2 & .... & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{array}$$

Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic

  f_{ij} = frequency of term i in document j

- May want to normalize *term frequency* (*tf*) across the entire corpus:

  $tf_{ij} = f_{ij} / max\{f_{ij}\}$

Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic.

  df_i = document frequency of term i

  = number of documents containing term i

  idf_i = inverse document frequency of term i,

  = $\log_2 (N/ df_i)$

  (N: total number of documents)

- Language model: An indication of a term's *discrimination* power.
  - Log used to dampen the effect relative to *tf*.

TF-IDF Weighting

- A typical weighting is *tf-idf weighting*:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Experimentally, *tf-idf* has been found to work well.

Computing TF-IDF: An Example

Given a document containing terms with given frequencies:

A(3), B(2), C(1)

Assume collection contains 10,000 documents and document frequencies of these terms are:

A(50), B(1300), C(250)

Then:

A: tf = 3/3; idf = log(10000/50) = 5.3; tf-idf = 5.3

B: tf = 2/3; idf = log(10000/1300) = 2.0; tf-idf = 1.3

C: tf = 1/3; idf = log(10000/250) = 3.7; tf-idf = 1.2

Query Vector

- Query vector is typically treated as a document and also tf-idf weighted.
- Alternative is for the user to supply weights for the given query terms.

Similarity Measure

- We now have vectors for all documents in the collection, a vector for the query, how to compute similarity?
- A similarity measure is a function that computes the *degree of similarity* between two vectors.
- Using a similarity measure between the query and each document:
  - It is possible to rank the retrieved documents in the order of presumed relevance.
  - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Desiderata for proximity

- If d_1 is near d_2, then d_2 is near d_1.
- If d_1 near d_2, and d_2 near d_3, then d_1 is not far from d_3.
- No document is closer to d than d itself.
  - Sometimes it is a good idea to determine the maximum possible similarity as the “distance” between a document d and itself

First cut: Euclidean distance

- Distance between vectors d_1 and d_2 is the length of the vector $|d_1 - d_2|$.
  - Euclidean distance
- Exercise: Determine the Euclidean distance between the vectors (0, 3, 2, 1, 10) and (2, 7, 1, 0, 0)
- Why not a great idea?
  - We still haven't dealt with the issue of length normalization
  - Long documents would be more similar to each other by virtue of length, not topic

Second Cut: Inner Product

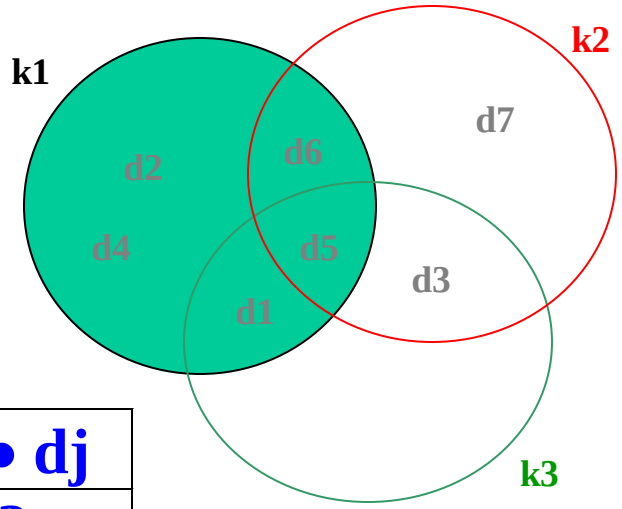
- Similarity between vectors for the document $\boldsymbol{d}_i$ and query $\boldsymbol{q}$ can be computed as the vector inner product:

$$\text{sim}(\boldsymbol{d}_j,\boldsymbol{q}) = \boldsymbol{d}_j \bullet \boldsymbol{q} = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$$

where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query

- For binary vectors, the inner product is the number of matched query terms in the document (size of intersection).
- For weighted term vectors, it is the sum of the products of the weights of the matched terms.

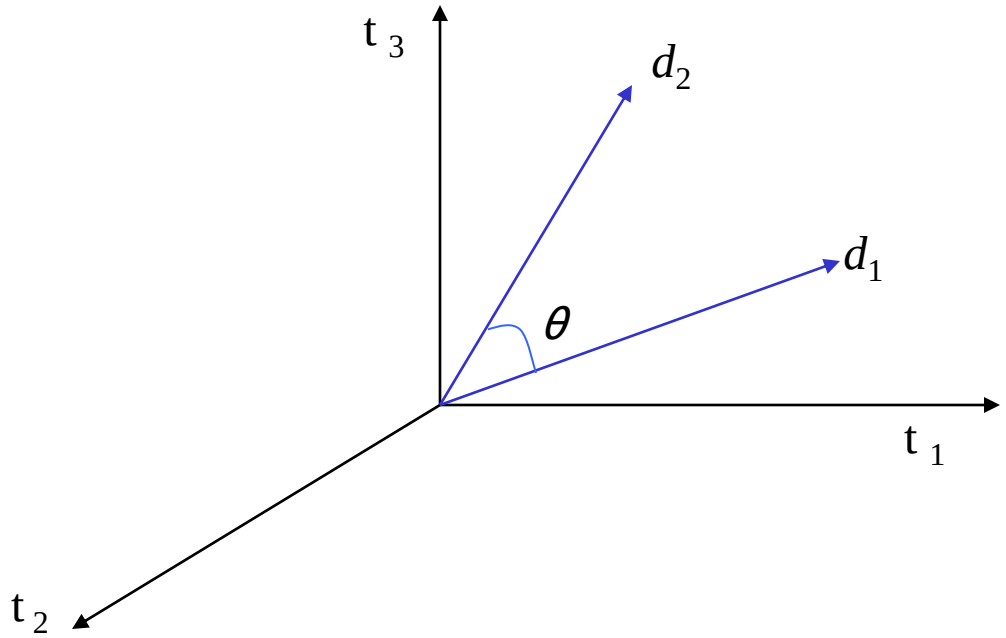
Inner Product: Example



	k1	k2	k3	q • dj
d1	1	0	1	2
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1
q	1	1	1	

Cosine Similarity

- Distance between vectors d_1 and d_2 *captured* by the cosine of the angle *x* between them.
- Note – this is *similarity*, not distance



Cosine Similarity

$$sim(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\left|\vec{d}_j\right|\left|\vec{d}_k\right|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

- Cosine of angle between two vectors
- The denominator involves the lengths of the vectors
- So the cosine measure is also known as the *normalized inner product*

$$\text{Length } \left|\vec{d}_j\right| = \sqrt{\sum_{i=1}^{n} w_{i,j}^2}$$

Example

- Documents: Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*

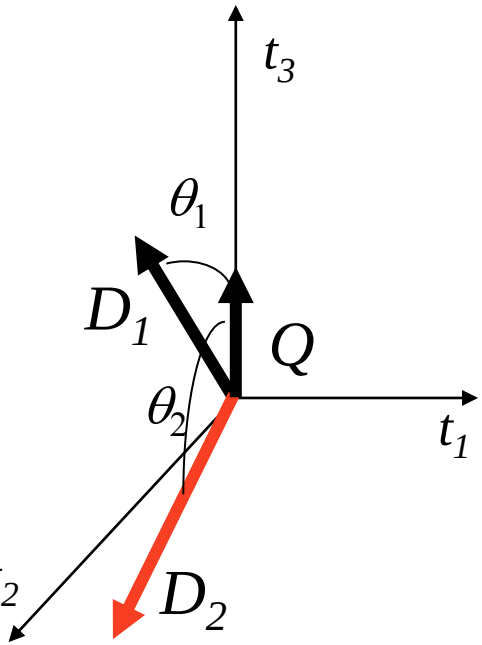
	SaS	PaP	WH
affection	**115**	**58**	**20**
jealous	**10**	**7**	**11**
gossip	**2**	**0**	**6**

	SaS	PaP	WH
affection	**0.996**	**0.993**	**0.847**
jealous	**0.087**	**0.120**	**0.466**
gossip	**0.017**	**0.000**	**0.254**

- cos(SAS, PAP) = .996 x .993 + .087 x .120 + .017 x 0.0 = 0.999
- cos(SAS, WH) = .996 x .847 + .087 x .466 + .017 x .254 = 0.929

Cosine Similarity *vs.* Inner Product

- Cosine similarity measures the cosine of the angle between two vectors.
- Inner product normalized by the vector lengths.



$$\text{CosSim}(\boldsymbol{d}_j, \boldsymbol{q}) = \frac{\vec{d}_j \cdot \vec{q}}{\left|\vec{d}_j\right| \cdot \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$

$$\text{InnerProduct}(\boldsymbol{d}_j, \boldsymbol{q}) = \vec{d}_j \cdot \vec{q}$$

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 10 / $\sqrt{(4+9+25)(0+0+4)}$ = 0.81
$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim(D_2 , Q) = 2 / $\sqrt{(9+49+1)(0+0+4)}$ = 0.13
$Q = 0T_1 + 0T_2 + 2T_3$

D_1 is 6 times better than D_2 using cosine similarity but only 5 times better using inner product.

Comments on Vector Space Models

- Simple, mathematically based approach.
- Can be applied to information retrieval and text classification
- Considers both local (*tf*) and global (*idf*) word occurrence frequencies.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses
- Allows efficient implementation for large document collections

Naïve Implementation

Convert all documents in collection D to tf-idf weighted vectors, $\boldsymbol{d}_j$, for keyword vocabulary V.

Convert query to a tf-idf-weighted vector $\boldsymbol{q}$.

For each $\boldsymbol{d}_j$ in D do

    Compute score $s_j = \text{cosSim}(\boldsymbol{d}_j, \boldsymbol{q})$

Sort documents by decreasing score.

Present top ranked documents to the user.

Time complexity: $O(|V| \cdot |D|)$ Bad for large V & D !

$|V| = 10{,}000$; $|D| = 100{,}000$; $|V| \cdot |D| = 1{,}000{,}000{,}000$

Practical Implementation

- Based on the observation that documents containing none of the query keywords do not affect the final ranking
- Try to identify only those documents that contain at least one query keyword
- Actual implementation of an inverted index

Step 1: Preprocessing

- Implement the preprocessing functions:
  - Tokenization
  - Stop word removal
  - Stemming

- Input: Documents that are read one by one from the collection
- Output: Tokens to be added to the index
  - No punctuation, no stop-words, stemmed

Step 2: Indexing

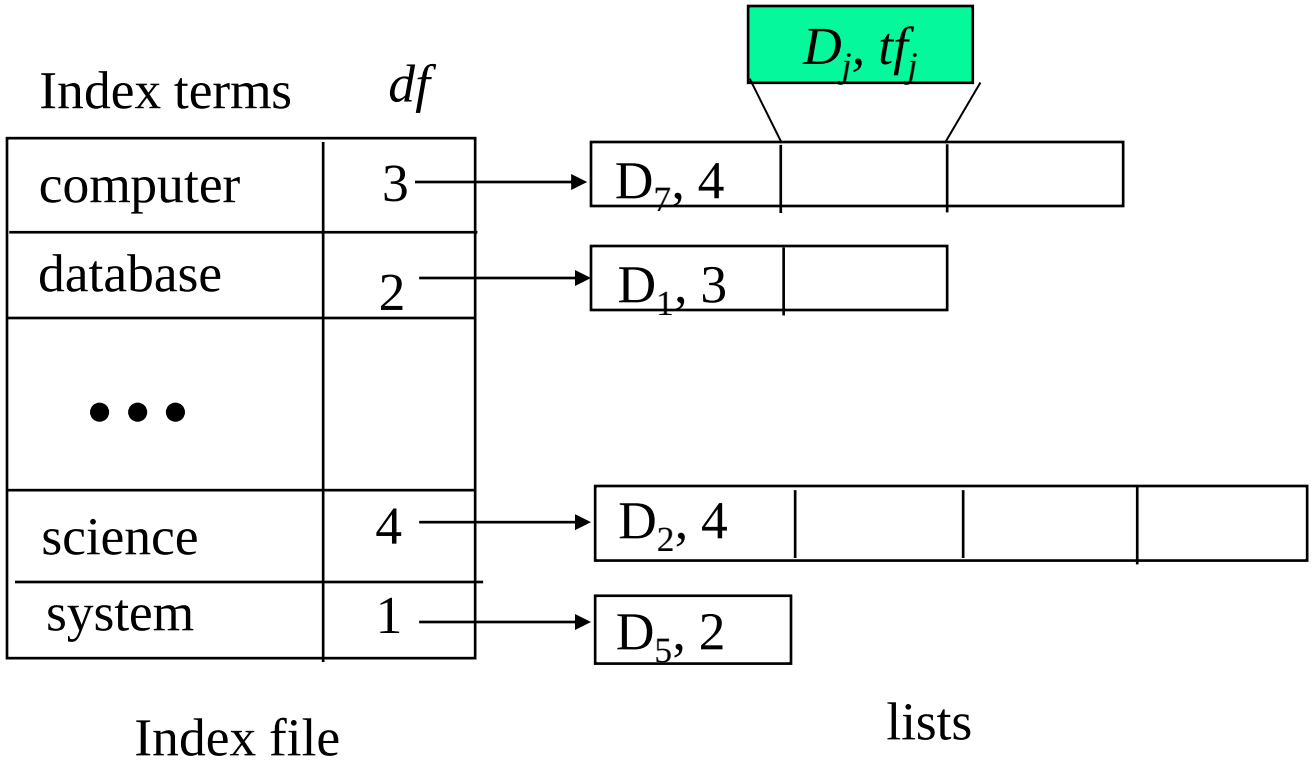
- Build an inverted index, with an entry for each word in the vocabulary

- Input: Tokens obtained from the preprocessing module
- Output: An inverted index for fast access

Step 2 (cont'd)

- Many data structures are appropriate for fast access
  - B-trees, skipped lists, hashtables
- We need:
  - One entry for each word in the vocabulary
  - For each such entry:
    - Keep a list of all the documents where it appears together with the corresponding frequency → TF
  - For each such entry, keep the total number of occurrences in all documents:
    - → IDF

Step 2 (cont'd)


D_j, tf_j
Index terms
df
computer
3
database
2
science
4
system
1
D_7, 4
D_1, 3
D_2, 4
D_5, 2
Index file
lists

Step 2 (cont'd)

- TF and IDF for each token can be computed in one pass
- Cosine similarity also required document lengths
- Need a second pass to compute document vector lengths
  - The length of a document vector is the square-root of sum of the squares of the weights of its tokens.
  - The weight of a token is TF * IDF
  - We must wait until IDF's are known (and therefore until all documents are indexed) before document lengths can be determined.
- Do a second pass over all documents: keep a list or hashtable with all document id-s, and for each document determine its length.

Step 3: Retrieval

- Use inverted index (from step 2) to find the limited set of documents that contain at least one of the query words.
- Incrementally compute cosine similarity of each indexed document as query words are processed one by one.
- To accumulate a total score for each retrieved document, store retrieved documents in a hashtable, where the document id is the key, and the partial accumulated score is the value.
- Input: Query and Inverted Index (from Step 2)
- Output: Similarity values between query and documents

Step 4: Ranking

- Sort the hashtable including the retrieved documents based on the value of cosine similarity
- Return the documents in descending order of their relevance

- Input: Similarity values between query and documents
- Output: Ranked list of documented in reversed order of their relevance

2. Joining the NLP world

Plugging into available resources for other languages

- **Automatic construction of parallel corpora**
- Construction of basic resources and tools based on parallel corpora:
  - translation models and bilingual lexicons
  - knowledge induction across parallel texts – POS tagging, word sense disambiguation
- Applications
  - statistical machine translation
  - cross-language information retrieval

Automatic construction of parallel corpora

Jiang Chen, Jian-Yun Nie

- Why parallel corpora?
  - machine translation
  - cross language information retrieval
  - …
- Existing parallel corpora:
  - the Canadian Hansards (English-French-Inuktitut (only local parliament))
  - the Hong Kong Hansards (English-Chinese)
  - …
  - see http://www.cs.unt.edu/~rada/wpt

Building a parallel corpus

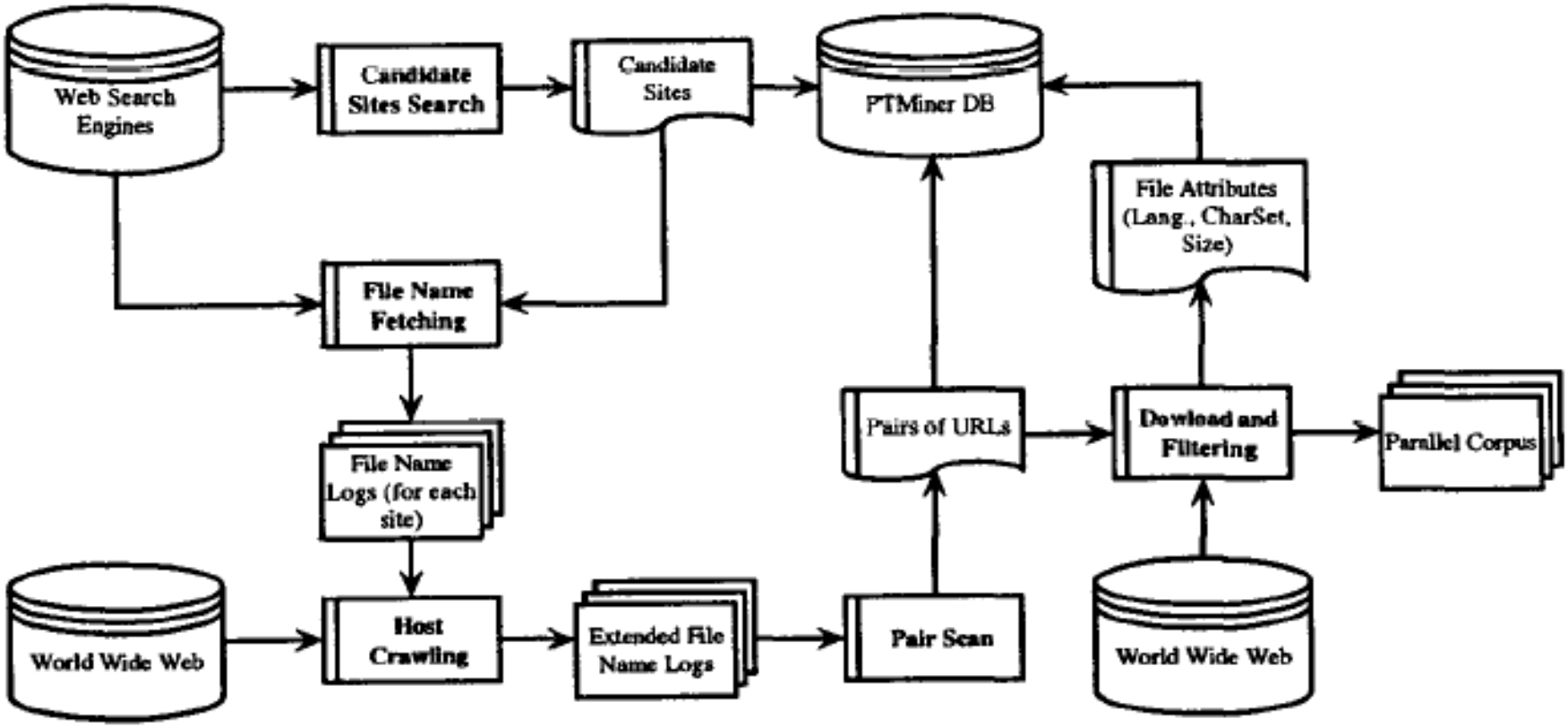
- Using the largest multi-lingual resource – the Web
  - search engines
  - anchor text for hyperlinks

Parallel text mining algorithm

- Search for candidate sites that may contain parallel pages;
  - anchor texts such as: “English version”, “in English”, …
- Fetch the URLs of Web pages indexed by the search engine;
- Follow these URLs and search each site separately for more URLs
- Scan for pairs from the URLs for each site
  - name patterns:
    - file-en.html file-ch.html
    - …/english/…/file.html …/chinese/…/file.html
- Download pages pairs, and filter out non-parallel ones by comparing file size, language and character sets, file structure

Automatic construction of parallel corpora

Jiang Chen, Jian-Yun Nie



Aligning Parallel Texts

- For similar languages:
  - cognates
  - syntactic structures
  - unambiguous words
  - dictionaries
- For very different languages:
  - HTML markup language
  - unambiguous words
  - dictionaries
  - see the tutorial on Machine Translation (Daniel Marcu)

2. Joining the NLP World

Plugging into available resources for other languages

- Automatic construction of parallel corpora
- Construction of basic resources and tools based on parallel corpora:
  - translation models and bilingual lexicons
  - **knowledge induction across parallel texts – POS tagging, word sense disambiguation**
- Applications
  - statistical machine translation
  - cross-language information retrieval

Part-of-speech Tagging Using Parallel Resources

- (Ngai & Yarowsky ‘01) (Borin ‘03)
- Assumption:

  “word pairs that are good translations of each other are likely to be the same parts of speech in their respective languages”

- Is it correct?

Parts-of-speech in Languages of the World

- Nouns and verbs are the only universal parts of speech
  - it seems like verbs are not all that necessary
- In closely related languages, we can assume that the same POS exist
- But in translation, not all POS are equally likely to remain invariant
- If we know which POS change in translation between a pair of languages L_1 and L_2, then we can use this knowledge, and a POS tagger for L_1 to tag L_2

Investigating the German-Swedish pair

- **word align** a Swedish-German parallel text (40% recall)
- **POS tag** the German text with a POS tagger (Morphy)
- **assign a the POS** of a German word to its Swedish counterpart (if an alignment exists)
- **assess the accuracy of the POS tags** assigned in the previous step.

POS Category Correspondences

- Categories: nouns, verbs, …

Correct alignments (64 of 78)		**Incorrect alignments (14 of 78)**	
correct POS 61 (95%)	incorrect POS 3 (5%)	correct POS 1 (8%)	incorrect POS 13 (92%)

  - correct alignments and correct POS go hand in hand
  - POS tag subcategories (e.g. inflectional information) were in general not relevant with the exception of number: the German value was the correct choice for the Swedish counterpart 27 times out of 29

What can We Conclude?

- Using word alignment as a stand-in for, or as a complement to, POS tagging is useful and worth exploring further
- Prerequisites:
  - the languages should be genetically close;
  - high word alignment precision needed;
  - only coarse POS tagging (main category, not fine morphosyntactic distinctions) seems possible

Sense Discrimination Using Parallel Texts

- There is controversy as to what exactly is a “word sense” (e.g., Kilgarriff, 1997)
- It is sometimes unclear how fine grained sense distinctions need to be to be useful in practice.
- Parallel text may present a solution to both problems!
  - Text in one language and its translation into another
- Manual annotation of sense tags is not required! However, text must be word aligned (translations identified between the two languages).

Word Senses as Word Translations

- Resnik and Yarowsky (1997) suggest that word sense disambiguation concern itself with sense distinctions that manifest themselves across languages.
  - A “bill” in English may be a “pico” (bird jaw) in or a “cuenta” (invoice) in Spanish.
- Given word aligned parallel text, sense distinctions can be discovered. (e.g., Li and and Li, 2002, Diab, 2002)
- See the tutorial on WSD and cross-lingual sense distinctions (Ide & Tufis)

2. Joining the NLP world

Plugging into available resources for other languages

- Automatic construction of parallel corpora
- Construction of basic resources and tools based on parallel corpora:
  - translation models and bilingual lexicons
  - knowledge induction across parallel texts – POS tagging, word sense disambiguation
- Applications
  - **statistical machine translation**
  - cross-language information retrieval

Statistical Machine Translation

- Translation systems based on:
  - Translation models learned from parallel corpora
  - Language models learned from monolingual corpora
- Translation quality depends on amount of data available
- See the tutorial on statistical machine translation (Daniel Marcu)

2. Joining the NLP world

Plugging into available resources for other languages

- Automatic construction of parallel corpora
- Construction of basic resources and tools based on parallel corpora:
  - translation models and bilingual lexicons
  - knowledge induction across parallel texts – POS tagging, word sense disambiguation
- Applications
  - statistical machine translation
  - **cross-language information retrieval**

Applications: cross-language information retrieval

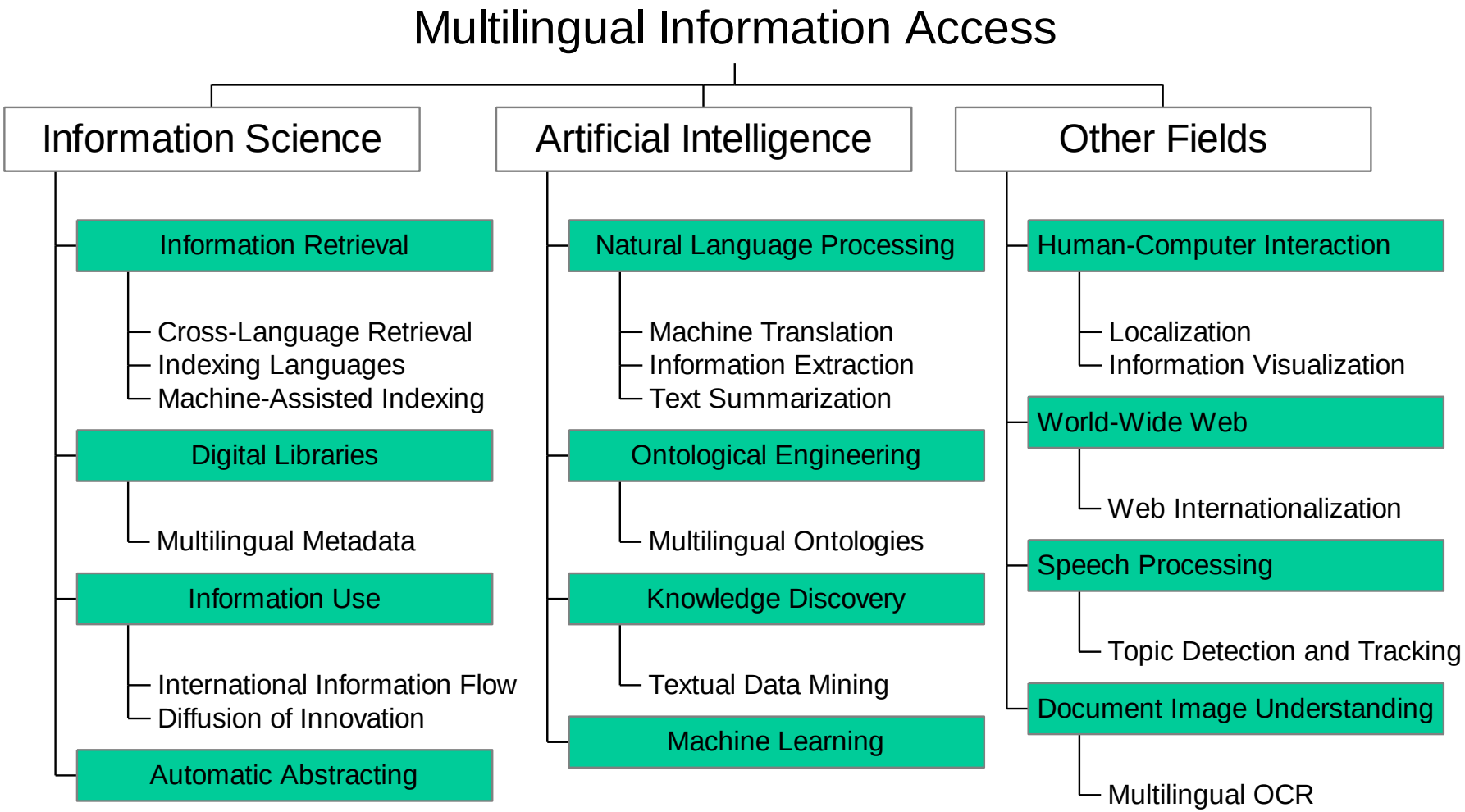
- Information retrieval:
  - documents
  - queries
- Cross language information retrieval
  - translate documents
  - translate queries

The General Problem

Find documents written in any language

– Using queries expressed in a single language

Multilingual information access



Why Do Cross-Language IR?

- When users can read several languages
  - Eliminates multiple queries
  - Query in most fluent language
- Monolingual users can also benefit
  - If translations can be provided
  - If it suffices to know that a document exists
  - If text captions are used to search for images

Supply Side: Internet Hosts

English	33,878,764	.com .net .edu .us .mil .uk .ca .au .org .gov .nz .ie
Japanese	1,686,534	.jp
German	1,684,396	.de .at .ch
French	653,916	.fr .be
Dutch	564,129	.nl
Finnish	546,244	.fi
Spanish	473,422	.es .mx .ar .cl .co .uy
Chinese	458,509	.tw .hk .sg .cn
Swedish	431,809	.se

Guess – What will be the most widely used language on the Web in 2010?

Source: Network Wizards Jan 99 Internet Domain Survey

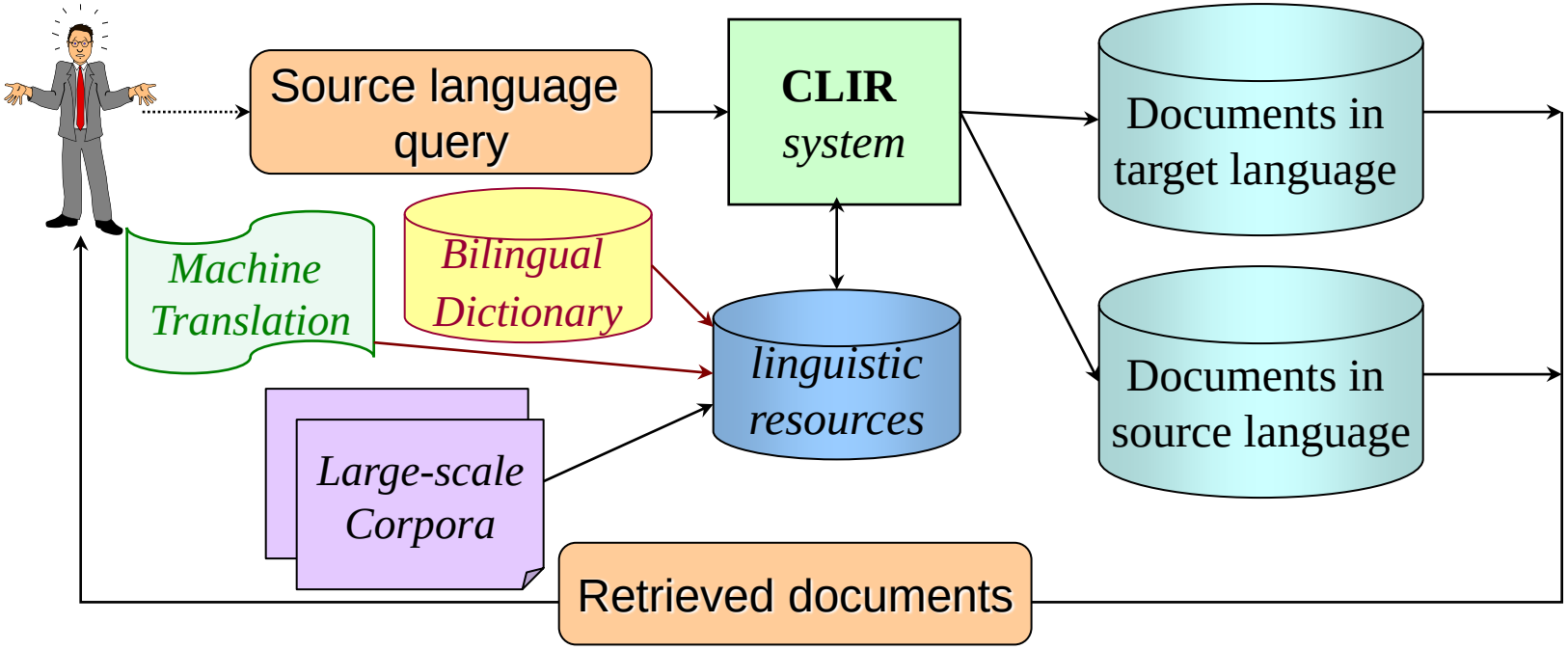
Demand Side: Number of Speakers

Chinese	885,000,000
English	450,000,000
Hindi-Urdu	333,000,000
Spanish	266,000,000
Portuguese	175,000,000
Bengali	162,000,000
Russian	153,000,000
Arabic	150,000,000
Japanese	126,000,000
French	122,000,000

Source: http://www.g11n.com/faq.html

Information retrieval

identify documents, which best match **users needs**, as expressed by the **query**



Documents Retrieval – Summarization -Translation, etc.

Search Technology

Language Identification

Chinese Feature Assignment

Monolingual Chinese Matching

1: 0.72
2: 0.48

Chinese Query

Chinese Feature Assignment

English Feature Assignment

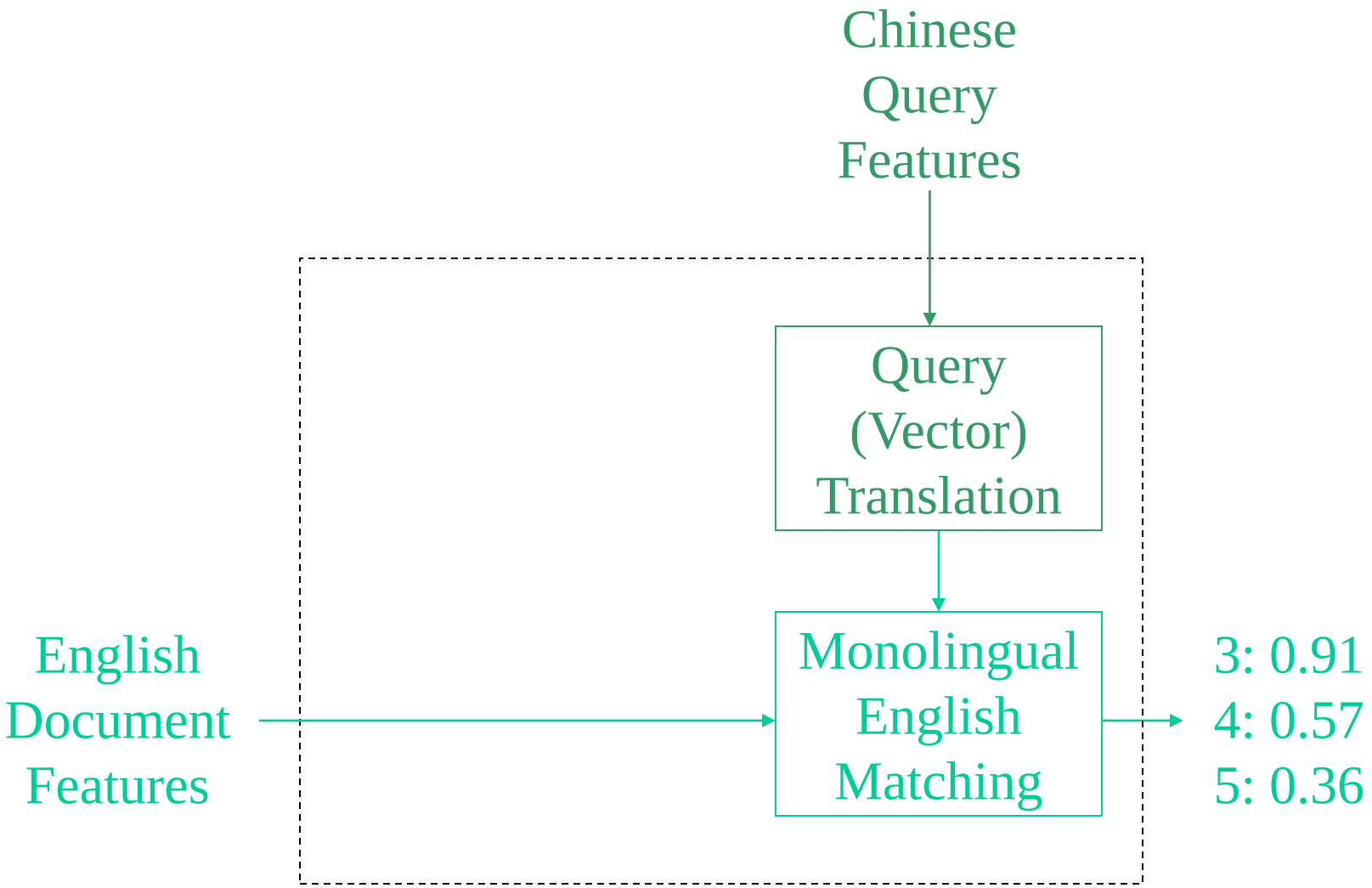
Cross-Language Matching

3: 0.91
4: 0.57
5: 0.36

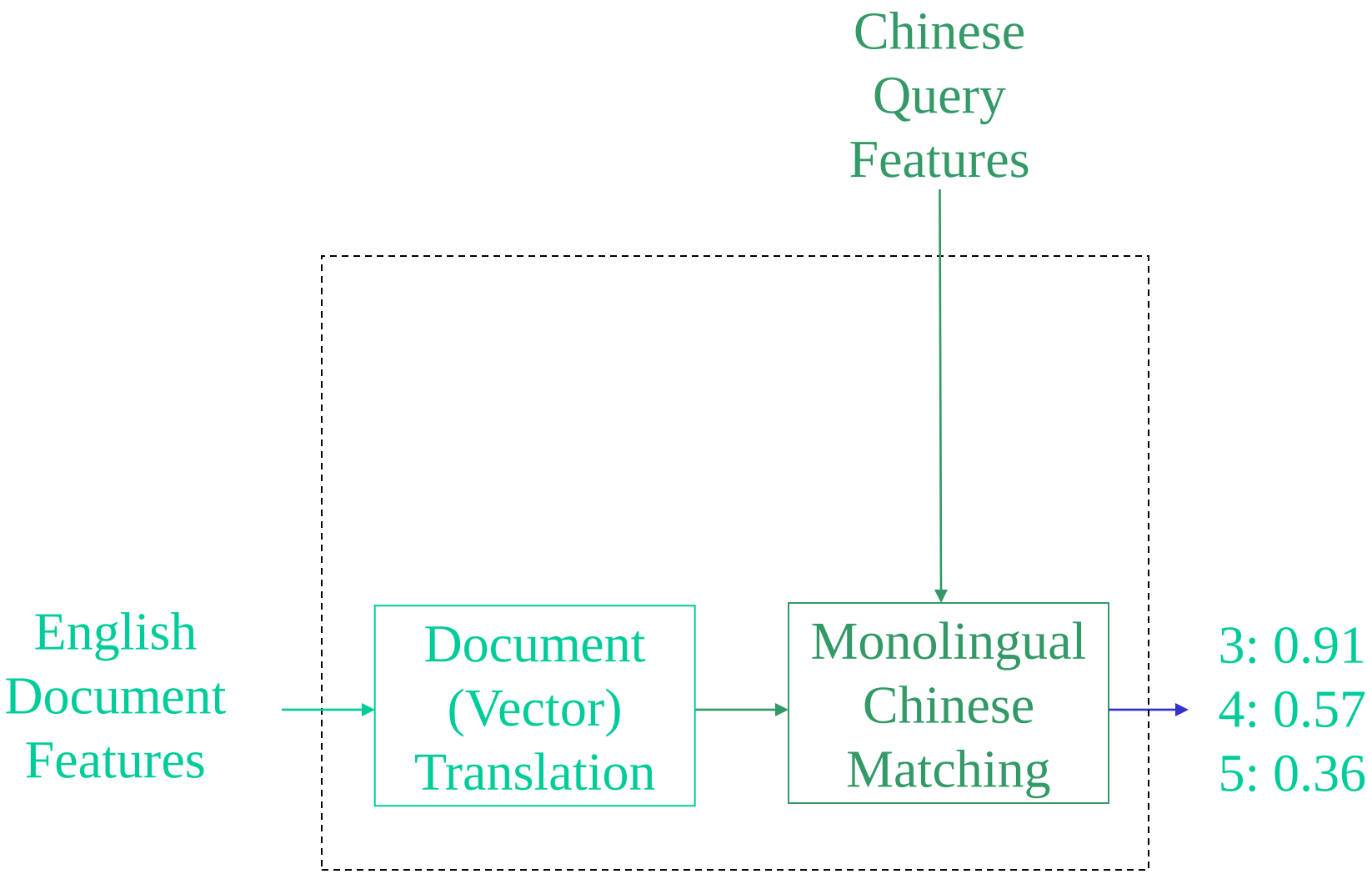
Design Decisions

- What to index?
  - Free text or controlled vocabulary
- What to translate?
  - Queries or documents
- Where to get translation knowledge?

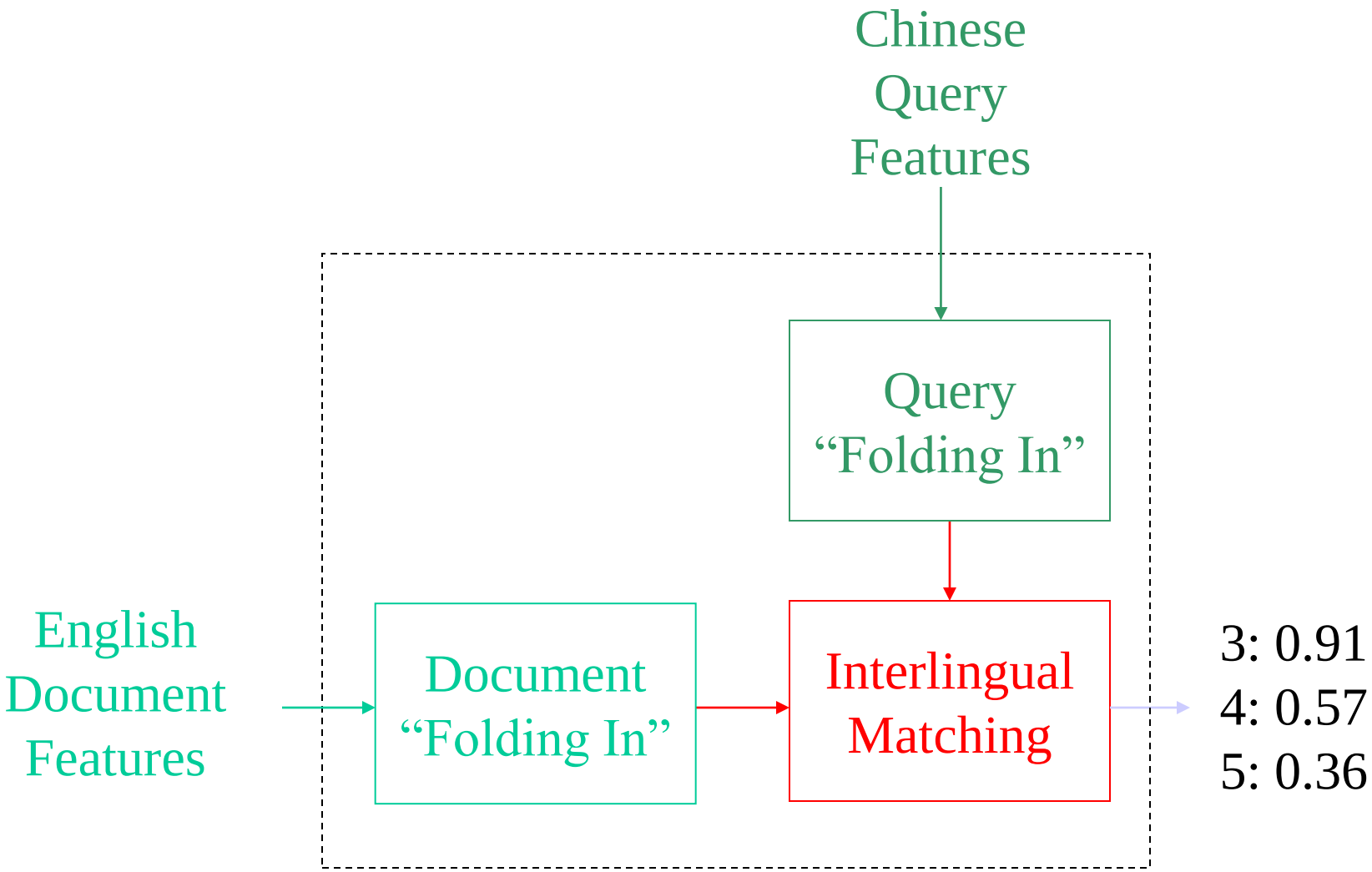
Query Vector Translation



Document Vector Translation



Matching Interlingual Representations



Query vs. Document Translation

- Query translation
  - Very efficient for short queries
    - Not as big an advantage for relevance feedback
  - Hard to resolve ambiguous query terms
- Document translation
  - May be needed by the selection interface
    - And supports adaptive filtering well
  - Slow, but only need to do it once per document
    - Poor scale-up to large numbers of languages

Translation Knowledge

- A lexicon
  - e.g., extract term list from a bilingual dictionary
- Corpora
  - Parallel or comparable, linked or unlinked
- Algorithmic
  - e.g., transliteration rules, cognate matching
- The user

Types of Lexicons

- Ontology
  - Representation of concepts and relationships
- Thesaurus
  - Ontology specialized for retrieval
- Bilingual lexicon
  - Ontology specialized for machine translation
- Bilingual dictionary
  - Ontology specialized for human translation

Multilingual Thesauri

- Adapt the knowledge structure
  - Cultural differences influence indexing choices
- Use language-independent descriptors
  - Matched to a unique term in each language
- Three construction techniques
  - Build it from scratch
  - Translate an existing thesaurus
  - Merge monolingual thesauri

Machine Readable Dictionaries

- Based on printed bilingual dictionaries
  - Becoming widely available
- Used to produce bilingual term lists
  - Cross-language term mappings are accessible
    - Sometimes listed in order of most common usage
  - Some knowledge structure is also present
    - Hard to extract and represent automatically
- The challenge is to pick the right translation

Unconstrained Query Translation

- Replace each word with every translation
  - Typically 5-10 translations per word
- About 50% of monolingual effectiveness
  - Ambiguity is a serious problem
  - Example: Fly (English)
    - 8 word senses (e.g., to fly a flag)
    - 13 Spanish translations (enarbolar, ondear, …)
    - 38 English retranslations (hoist, brandish, lift…)

Phrase Indexing

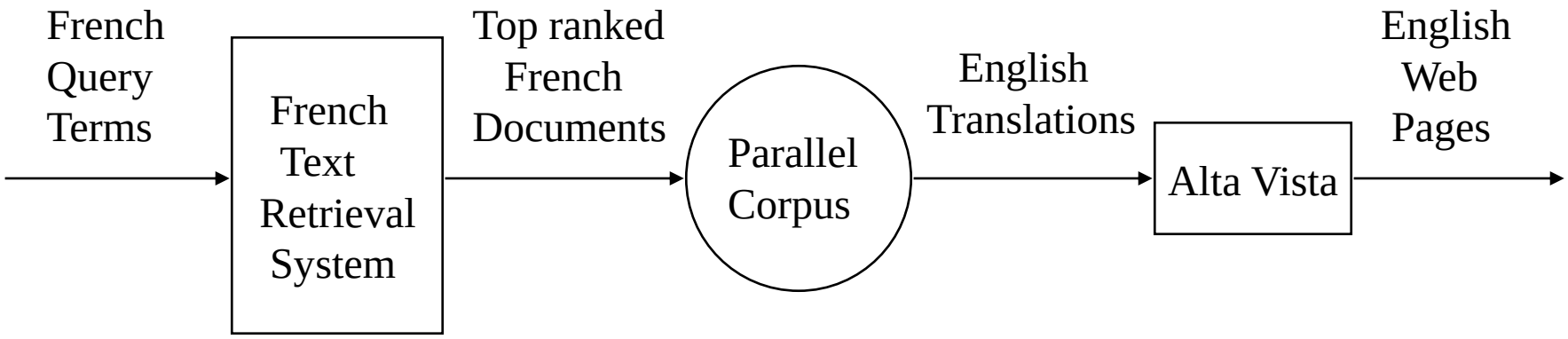
- Improves retrieval effectiveness two ways
  - Phrases are less ambiguous than single words
  - Idiomatic phrases translate as a single concept
- Three ways to identify phrases
  - Semantic (e.g., appears in a dictionary)
  - Syntactic (e.g., parse as a noun phrase)
  - Cooccurrence (words found together often)
- Semantic phrase results are impressive

Types of Bilingual Corpora

- Parallel corpora: translation-equivalent pairs
  - Document pairs
  - Sentence pairs
  - Term pairs
- Comparable corpora
  - Content-equivalent document pairs
  - E.g. newspaper articles in different languages, on the same day (for the same event)
- Unaligned corpora
  - Content from the same domain

Pseudo-Relevance Feedback

- Enter query terms in French
- Find top French documents in parallel corpus
- Construct a query from English translations
- Perform a monolingual free text search



Similarity-Based Dictionaries

- Automatically developed from aligned documents
  - Terms E1 and E3 are used in similar ways
    - Terms E1 & S1 (or E3 & S4) are even more similar
- For each term, find most similar in other language
  - Retain only the top few (5 or so)
- Performs as well as dictionary-based techniques
  - Evaluated on a comparable corpus of news stories
    - Stories were automatically linked based on date and subject

Exploiting Unaligned Corpora

- Documents about the same set of subjects
  - No known relationship between document pairs
  - Easily available in many applications

- Two approaches
  - Use a dictionary for rough translation
    - But refine it using the unaligned bilingual corpus
  - Use a dictionary to find alignments in the corpus
    - Then extract translation knowledge from the alignments

Conclusions I

- 7000 languages worldwide, but resources only for a few dozens. [Bird, S et al., 2009, Natural Languages Processing with Python: Analyzing Text with Natural Language Toolkit - http://victoria.lviv.ua/html/fl5/NaturalLanguageProcessingWithPython.pdf] http://www.ethnologue.com

Hopeless situation?

- No! The multilingual Web can provide the resources required to build basic tools for a new language “in one-person day”

Conclusions II

- Languages share similarities and have differences:

Why?

- ***because of similarities we can port algorithms from one language to the next***;

- ***differences show us how rich and wonderful languages are, and give us interesting research topics.***


ngiyabonga
شكراً جزيلاً
рахмат
danke
謝謝
spas
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
dank je
misaotra
matondo
paldies
grazzi
спасибі
blagodaram
kiitos
dankie
nanni
nandri
dhanyavad
thank you
gracias
хвала
asante
manana
obrigada
tenki
bedankt
bayarlalaa
gràçie
hvala
maururu
köszönöm
aciu
dankon
akun
djiere dieuf
tau
mochchakkeram
murakoze
enkosi
dziękuję
дякую
mamnun
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
chokrane
sobodi
dėkuji
sagolun
sukriya
kop khun krap
taiku
obrigado
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
arigatô
takk
dakujem
trugarez
kam sah hamnida
rahmat
terima kasih
rahmet
diolch
dhanyavadagalu
shukriya
mercê
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ
merci