Machine Translation

AIGS 537
What is Machine Translation (MT)?

- **Translation** is the process of:
  - Moving texts from one human language (**source language**) to another (**target language**),
  - In a way that preserves meaning

- **Machine translation (MT)** automates (part of) the process:
  - Fully-automatic translation
  - Computer-aided (human) translation

---

**Source text:**
قال الرئيس الأمريكي، جورج بوش، إنه يمكن إقامة دولة فلسطينية...

**Target text:**
The President of the USA, George Bush, said that there is a possibility that a Palestinian state may be established...
Google Translator

Machine translation, sometimes referred to by the abbreviation MT (not to be confused with computer-aided translation, machine-aided human translation (MAHT) or interactive translation), is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another.

On a basic level, MT performs mechanical substitution of words in one language for words in another, but that alone rarely produces a good translation because recognition of whole phrases and their closest counterparts in the target language is needed. Not all words in one language have equivalent words in another language, and many words have more than one meaning. In addition, two given languages may have completely different structures.

약어 MT로 지칭되는 기계 번역 (컴퓨터 보조 번역, 기계 보조 인간 번역 (MAHT) 또는 대화식 번역과 혼동하지 말 것)은 번역을위한 소프트웨어의 사용을 조사하는 전산 언어학의 하위 풀드임니다. 한 언어에서 다른 언어로 텍스트 또는 음성.

기본적으로 MT는 한 언어의 단어를 다른 언어의 단어로 기계적으로 대체하지만, 전체 문구와 대상 언어에서 가장 가까운 대응어를 인식해야하기 때문에 좋은 번역을 거의 만들지 않습니다. 한 언어의 모든 단어가 다른 언어의 동등한 단어를 갖는 것은 아니며 많은 단어가 동일한 의미를 갖습니다. 또한, 주어진 두 언어는 완전히 다른 구조를 가질 수 있습니다.

yag-eo MT로 지칭되는 기계 번역 (컴퓨터 보조 번역, 기계 보조 인간 번역 (MAHT) 또는 대화식 번역과 혼동하지 말 것)은 번역을위한 소프트웨어의 사용을 조사하는 전산 언어학의 하위 풀드임니다. 한 언어에서 다른 언어로 텍스트 또는 음성.

자세히
Different Scenarios for MT

**Assimilation** (acquisition)
- Many SLs, one TL
- All-purpose translation
  - Any style, topic
  - Partial analysis
  - Post-editing

➤ Requirements for MT:
- Robustness
- Coverage

**Dissemination** (distribution)
- One SL, many TLs
- Special translation
  - Controlled style, single topic
  - Full analysis
  - No post-editing

➤ Requirements for MT:
- Textual quality
MT Problems: Analysis Ambiguity

- Lexical ambiguity
  - *Stay away from the bank.*

- Syntactic ambiguity
  - *John hit the dog with the stick.*

- Anaphoric ambiguity
  - *My cat was chasing a mouse. It played with it.*
MT Problems: Divergence/Mapping

- Different word orders
  - SOV (44.78%) – Korean, Japanese, Turkish, Hindi, ...
  - SVO (41.79%) – English, French, Chinese, Thai, ...
  - VSO (9.20%) – Arabic, Hebrew, ...

- Syntactic structure is not preserved across translation
  - Example: N ↔ M, Head-switching
    - English: *The bottle floated into the cave.*
    - Spanish: *La botella entró a la cueva flotando.*
      (Lit: the bottle entered the cave floating)

- Example: SUBJ ↔ OBJ
  - English: *I like Mary.*
  - Spanish: *Maria me gusta.* (Lit: Mary me-ACC please)
MT Problems: Generation Ambiguity

- **Lexical selection**
  - English: *wear*
  - Korean: 입다 (generic), 신다 (shoes),끼다 (ring, gloves), etc.

- **Tense generation**
  - *Mary went to Mexico. During her stay she learned Spanish.*
    - *went* ➔ Spanish: *iba* (simple past/preterite)
  
  - *Mary went to Mexico. When she returned she started to speak Spanish.*
    - *went* ➔ Spanish: *fue* (ongoing past/imperfect)
MT Problems: Different Styles

- Different way of thinking/representing concepts
  - English: *The key opened the door.*
    → Korean: ?

- Different lexical choice (cultural context)
  - English: *wife*
    → Japanese: 家内 (my wife), 奥さん (other people’s wife)
    → Korean: ?

- Different lexical choice (situational context)
  - Field (subject matter): technical (legal, medical, ...), literary, ...
  - Mode (medium of text): news article, report, email, instant messages
  - Tenor (degree of formality): formal, casual, intimate
The Vauquois Triangle (1968)

Bernard Vauquois was an influential leader in MT research in the 1970s at Grenoble University (at RBMT era)
Research Paradigms of MT

- **Information** components to be used for translation
  - Linguistics-based (rule-based)
    - Direct / Transfer / Interlingua MT
  - Corpus-based
    - Example-based, Statistics-based, Neural MT
  - Hybrid

- **Pairs & directions** of languages
  - Bilingual
  - Multilingual

- **Human involvement** in translation
  - Batch (MT)
  - Interactive (HAMT)
  - Human in full control (MAHT)
Direct MT (Jp → En)

- Input: *watashihatsukuenouenopenwojonniageta.* (私は机の上のペンをジョンにあげた。)

<table>
<thead>
<tr>
<th>Stage</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Morphological analysis</td>
</tr>
<tr>
<td>2</td>
<td>Lexical transfer of content words</td>
</tr>
<tr>
<td>3</td>
<td>Various work relating to prepositions</td>
</tr>
<tr>
<td>4</td>
<td>SVO rearrangement</td>
</tr>
<tr>
<td>5</td>
<td>Local-word-order adjustment &amp; Miscellany</td>
</tr>
<tr>
<td>6</td>
<td>Morphological generation</td>
</tr>
</tbody>
</table>

1. *watashi ha tsukue no ue no pen wo jon ni ageru PAST.*
2. *I ha desk no ue no pen wo John ni give PAST.*
3. *I ha pen on desk wo John to give PAST.*
4. *I give PAST pen on desk John to.*
5. *I give PAST the pen on the desk to John.*
6. *I gave the pen on the desk to John.*
Direct MT: Local Word-order Adjustment

- Using MFOLT (Modality-Feature Ordering & Lexicalizing Table)
- Example:
  - Input: 彼は会議に出席できませんでした。(He could not attend the meeting.)
  - Output: 그는 회의에 출석할 수 없었습니다.

  ![Diagram showing the process of adjusting word order using MFOLT.](image-url)
Transfer MT (E→Hebrew)

- **Input:** *The blue car*

**Analysis**

- **NP**
  - **NP**
    - **The** [Art]
    - **Blue** [Adj]
    - **Car** [N]

**Rules:**

- **NP → [Adj] [N]**
- **NP → [Art] [NP]**
- **Art+NP → [N] [Adj]**
- **NP → [Art+NP]**
- **NP → [Art+N] [Art+Adj]**

**Transfer**

- **NP**
  - **NP**
    - **The** [Art]
    - **Car** [N]
    - **Blue** [Adj]

**Generation**

- **NP**
  - **NP**
    - **The** [Art]
    - **Car** [N]
    - **Blue** [Adj]
Interlingua-Based MT

- Works in 2 phases: *analysis & generation*
  - Analysis
    - Extracts the meaning of SL sentence in a form of IL (interlingua)
  - Generation
    - Generates TL sentence from the extracted IL

- Much more appropriate for multi-lingual MTs than transfer method
Interlingua MT (En → Sp)

- English: *I stabbed John.*
  → Spanish: *Yo le di puñaladas a Juan.* (Lit. I gave knife-wounds to John)
Example-Based MT (En → Kr)

- DB of translation examples

- Input sentence: My sister eats potatoes. → 나의 누나는 감자를 먹는다.

- Output sentence: 나의 누나는 감자를 먹는다.
Statistical MT

Untranslated ancient Egyptian language

French linguist J. Champollion

Rosetta Stone

Parallel text (in 3 languages)

SMT
Statistical MT: Word Alignment

- **English:**
  - null *The quick fox jumps over the lazy dog*

- **French:**
  - *Le renard rapide saut par-dessus le chien parasseux*
Statistical MT: Extracting Phrase Pairs

- Choose the most probable alignment
  - Viterbi alignment
- Collect all consistent phrase pairs from Viterbi path
Statistical MT: Phrase Translation Table

- Main knowledge source
  - Store phrase pairs \((f, \bar{e})\) with their probability \(\phi(f | \bar{e})\)

| German | \(f\) | English | \(\bar{e}\) | Probability | \(\phi(f | \bar{e})\) |
|--------|-------|---------|-------------|-------------|-----------------|
| der    | the   |          |             | 0.3         |                 |
| das    | the   |          |             | 0.4         |                 |
| das    | it    |          |             | 0.1         |                 |
| das    | this  |          |             | 0.1         |                 |
| ist    | is    |          |             | 1.0         |                 |
| das ist| it is |          |             | 0.2         |                 |
| das ist| this is |       |             | 0.8         |                 |
| es ist | it is |          |             | 0.8         |                 |
| es ist | this is |       |             | 0.2         |                 |
| ...    | ...   |          |             | ...         |                 |
Statistical MT: Phrase-Based Translation

- Three steps for generating $F$ from $E$:
  - Given $E = e_1, e_2, ..., e_n$

1. **Group** the English words into phrases $\bar{e}_1, \bar{e}_2, ..., \bar{e}_l$
2. **Translate** each English phrase into a foreign phrase
3. **Optionally reorder** the Foreign phrases

Diagram:
- **Grouping:**
  - English: tomorrow, I will fly, to the conference, in Canada
  - Phrases: tomorrow, I, will fly, to the conference, in Canada

- **Translating:**
  - German: morgen, ich, fliege, zur Konferenz, nach Kanada

- **Reordering:**
  - German: morgen, fliege, ich, nach Kanada, zur Konferenz

Hidden phrase boundaries
Classical ML vs. Deep Learning

Classical ML-based NLP

Deep Learning-based NLP (Black box)
Types of Neural Sequence Models

- A wide spectrum of sequence models with NNs ...

<table>
<thead>
<tr>
<th>One to One</th>
<th>One to Many</th>
<th>Many to One</th>
<th>Many to Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: No sequence</td>
<td>Output: No sequence (e.g.) Im2Caption</td>
<td>Input: Sequence (e.g.) Sentence classification, Multiple-choice QA</td>
<td>Input: Sequence (e.g.) MT, Video captioning, Open-ended QA</td>
</tr>
</tbody>
</table>

“Standard” classification
RNN Encoder-Decoder

- MT as a general **sequence-to-sequence** transduction
  - To model the probability \( P(E|F) \) of the output \( E \) given the input \( F \)

![Diagram of RNN Encoder-Decoder]

- Encoder
- Decoder

\[ \text{Encoder} \]

\[ \text{Decoder} \]

- RNN Encoder-Decoder
- MT as a general **sequence-to-sequence** transduction
- To model the probability \( P(E|F) \) of the output \( E \) given the input \( F \)
RNN Encoder-Decoder: 2 Hidden Layers
Encoding: 1\textsuperscript{st} Layer

Encoder 1\textsuperscript{st} layer
Encoding: 2\textsuperscript{nd} Layer

Encoder 2\textsuperscript{nd} layer
Decoding: 1\textsuperscript{st} Layer
Decoding: 2\textsuperscript{nd} Layer
Decoding: Word Probability Distribution

- Hidden states $\rightarrow$ scores $\rightarrow$ probabilities

- Feed the most likely word

- Beam-search decoder
  - Greedy search (if width = 1)
Sampling: Greedy Search

- A search graph with a vocabulary of \{a, b, </s>\}
  - Greedy search fails
Sampling: Beam Search (width = 2)

- A search graph with a vocabulary of \{a, b, </s>\}
  - Beam search succeeds
Questions?
What is Information Retrieval (IR)?

- Finding *needed information* (based on its *semantic* content) in a large collection of data

- Historically, IR is about *document retrieval*
  - Retrieving *documents relevant to user query* from a document collection
  - You may want to find:
    - *Give me information on the history of the Kennedys.*
      → An article about the Kennedys (*Text Retrieval*)
    - *What does a brain tumor look like on a CT-scan?*
      → A picture of a brain tumor (*Image Retrieval*)
    - *It goes like this: hmm hmm hahmmm . . .*
      → A certain song (*Music retrieval*)
Web Search Overview

1. Web crawling
2. Indexing
3. Searching

The Web

Indexes

Ad indexes

User

CG Appliance Express
Discount Appliances (650) 756-3931
www.cgappliance.com
San Francisco-Oakland-San Jose, CA

Miele Vacuum Cleaners
Miele Vacuums- Complete Selection
Free Shipping!
www.vacuums.com

Miele Vacuum Cleaners
-Miele- Free Air shipping!
All models. Helpful advice.
www.best-vacuum.com
Inverted Index (Inverted File)

- For each term $t$, we must store a list of all documents that contain $t$
- **Inverted Index = Dictionary + Postings**
  - Dictionary (Lexicon, Vocabulary, Index) – commonly kept in memory
  - Variable-size postings lists – stored on disk

```
Brutus
1 2 4 11 31 45 173 174

Caesar
1 2 4 5 6 16 57 132 ...

Calpurnia
2 31 54 101

Dictionary

Postings
```
Indexing Techniques

- **N-gram-based indexing**
  - N-grams: *connect* → *con, onn, nne, nec, ect*
  - Language independent
  - Easy to develop
  - Cannot handle synonyms
    - *car, auto, automobile*

- **Word-based indexing**
  - Can handle morphological variations
    - *connects, connected, connecting* → *connect*
  - Needs morphological analyzer
    - Cannot handle unknown words
    - Not suitable for news domain (too many named entities)
Basic Indexing Pipeline

Documents to be indexed

Tokenizer

Token stream

Linguistic modules

Modified tokens

Indexer

Inverted index

Friends, Romans, countrymen.

Text Processing

Friend  Roman  Countryman

friend  roman  countryman
Text Processing: Tokenization

- Task of **segmenting** a character sequence into pieces, called “tokens”
  - Tokens are candidates for index terms, after further processing

- Simple approach:
  - Chop on whitespace
  - Throw away punctuation characters

- Example:
  - Input: *Friends, lend me your ears;*
  - Output: 
    
    
    
    
    
    
    
    

- Major question: What are **valid tokens** to emit?
Text Processing: Normalization

- Transforming tokens (in indexed text as well as query words) into a **standard form**
  - Example: We want to match
    - *U.S.A.* → *USA*
    - *connects, connected, connecting* → *connect*
  - A “term” is a **normalized word**, which is an **entry in IR dictionary**
    - Equivalence class

- How to reduce **inflectional** (sometimes derivational) forms to a **common base form**
  - **Lemmatization** – Linguistic process (based on vocabulary & morphological analysis)
  - **Stemming** – Crude heuristic process
Text Processing: Stemming

- Crude heuristic process that chops off the word endings
  - Hoping to achieve what “principled” lemmatization attempts to do with a lot of linguistic knowledge
  - Language dependent

- Often strange-looking output

- Fortunately, works quite well for English

for example compressed and compression are both accepted as equivalent to compress

for example compress and compress are both accepted as equivalent to compress
Phrase Queries

- Want to answer a query “black sea” – as a phrase
  - Thus, the following sentence shouldn’t be a match:
    - “A man in black took a view of the sea”

- Posting in a positional index
  - \( \text{docID} + \text{a list of positions} \) in document
  - Sample format:

\[
\text{term, document freq : } \begin{cases} 
< \text{doc1, term freq : } < \text{position1, position2, } \ldots >; \\
\text{doc2: term freq : } < \text{position1, position2, } \ldots >; \\
\ldots >
\end{cases}
\]
Positional Index: Example

Example query: \( "\text{to}_1 \text{ be}_2 \text{ or}_3 \text{ not}_4 \text{ to}_5 \text{ be}_6" \)

- **to**, 993427:
  - \(< 1, 6: < 7, 18, 33, 72, 86, 231 >;\)
  - \(< 2, 5: < 1, 17, 74, 222, 255 >;\)
  - \(< 4, 5: < 8, 16, 190, 429, 433 >;\)
  - \(< 5, 2: < 363, 367 >;\)
  - \(< 7, 3: < 13, 23, 191 >; \ldots >\)

- **be**, 178239:
  - \(< 1, 2: < 17, 25 >;\)
  - \(< 4, 5: < 17, 191, 291, 430, 434 >;\)
  - \(< 5, 3: < 14, 19, 101 >; \ldots >\)

- **or**, .....  
- **not**, .....  

Phrase Query Processing

(1) Identify docs that contain all query terms
(2) Check positions

\[ \text{docID} = 4 \]
Index Construction: 3 Basic Steps

- **Step-1 (Parsing):**
  - Parse the collection into a list of terms, assembling all \(<term, docID>\) pairs
    - Recognition of the individual words (tokenization)
    - Elimination of stop words
    - Normalization (stemming, lemmatization)

- **Step-2 (Sorting):**
  - Sort the list of terms: location order → alphabetic order
    - Inverted list of \(<term-docID>\) pairs

- **Step-3 (Splitting):**
  - Split the Inverted list into two pieces:
    - Dictionary + Postings lists
  - Compute statistics like term & document frequencies
Index Construction: Parsing

- **Parse** the documents to extract words with their locations (*docIDs*)
  - A list of `<term, docID>` pairs (“*postings*”)

<table>
<thead>
<tr>
<th>Term</th>
<th>doc_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>was</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>i'</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
</tr>
<tr>
<td>brutus</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
</tr>
<tr>
<td>so</td>
<td>2</td>
</tr>
<tr>
<td>let</td>
<td>2</td>
</tr>
<tr>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>with</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>2</td>
</tr>
<tr>
<td>noble</td>
<td>2</td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
</tr>
<tr>
<td>hath</td>
<td>2</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

**Doc 1**

*I did enact Julius
Caesar I was killed
i' the Capitol;
Brutus killed me.*

**Doc 2**

*So let it be with
Caesar. The noble
Brutus hath told you
Caesar was ambitious*
Index Construction: Sorting

- **Sort** (invert) the list of `<term, docID>` pairs
  - Primary key: `term`
  - Secondary key: `docID`

We focus on this sort step. We have 100M items to sort.
Index Construction: Splitting

- **Split** the Inverted list into two:
  - **Dictionary**
  - **Postings lists**

<table>
<thead>
<tr>
<th>Term</th>
<th>doc_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambitious</td>
<td>2</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>brutus</td>
<td>1</td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>hath</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>i'</td>
<td>1</td>
</tr>
<tr>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>let</td>
<td>2</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
</tr>
<tr>
<td>noble</td>
<td>2</td>
</tr>
<tr>
<td>so</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>2</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc. freq</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambitious</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>be</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>did</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>hath</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>i'</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>it</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>let</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>me</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>noble</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>so</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>

Postings (docID)

2
1
1
1
2
1
2
1
2
1
2
2
1
2
1
2
2
1
Retrieval: Vector Space Model

- Each doc can now viewed as a **real-valued vector** of \( \text{tf} \times \text{idf} \) weights

- So, we have a \( |V| \)-dimensional **vector space**:
  - Terms are axes of the space (**basis vectors**)
  - Everything (docs, queries, terms) are **points** or **vectors** in this space
    - Very sparse vectors – most entries are zero
  - Very high-dimensional
    - May have 200,000+ dimensions (even with stemming)
Vector Space Proximity

- Which of $d_1$ and $d_2$ is more similar to $q$?
- How to measure the degree of similarity?
  - Distance, Angle, or Projection

\[ d_1 = 2t_1 + 3t_2 + 5t_3 \]
\[ d_2 = 3t_1 + 7t_2 + 1t_3 \]
\[ q = 0t_1 + 0t_2 + 2t_3 \]
Cosine Similarity

- So the matching score (cosine similarity) of a document $d_j$ with regard to a query $q$ is given by:

$$\text{sim}(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

- The length $|\vec{x}|$ is used for normalization purposes
  - Every matching-score is between 0 and 1

- The vectors are made up of $tf \times idf$ weights
Cosine Similarity: Example

- Let the weighted term vectors be:
  - \( d_1 = (2, 3, 5) \quad d_2 = (3, 7, 1) \quad q = (0, 0, 2) \)

- Then, cosine similarity scores are

\[
sim(d_1, q) = \frac{10}{\sqrt{(4 + 9 + 25)} \sqrt{(0 + 0 + 4)}} = 0.81
\]

\[
sim(d_2, q) = \frac{2}{\sqrt{(9 + 49 + 1)} \sqrt{(0 + 0 + 4)}} = 0.13
\]

- The doc \( d_1 \) is 6 times better than \( d_2 \) (in terms of cosine similarity)
- But, only 5 times better (in terms of inner product)
Many Variants in \( tf \times idf \) Weighting

<table>
<thead>
<tr>
<th>Term frequency</th>
<th>Document frequency</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>n (natural)</td>
<td>( tf_{t,d} )</td>
<td>n (no)</td>
</tr>
<tr>
<td>l (logarithm)</td>
<td>( 1 + \log(tf_{t,d}) )</td>
<td>t (idf) ( \log \frac{N}{df_t} )</td>
</tr>
<tr>
<td>a (augmented)</td>
<td>( 0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})} )</td>
<td>p (prob idf) ( \max{0, \log \frac{N - df_t}{df_t}} )</td>
</tr>
<tr>
<td>b (boolean)</td>
<td>\begin{cases} 1 &amp; \text{if } tf_{t,d} &gt; 0 \ 0 &amp; \text{otherwise} \end{cases}</td>
<td></td>
</tr>
<tr>
<td>L (log ave)</td>
<td>( \frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}<em>{t \in d}(tf</em>{t,d}))} )</td>
<td></td>
</tr>
</tbody>
</table>

Columns headed ‘x’ are acronyms for weight schemes

- Most popular one is *logarithmic* one
- According to [Zobel & Moffat, 98], there is no big difference in terms of quality for most \( tf \times idf \) heuristics
Vector Space Model: Examples

- **SMART** (1964, 1971)
  - By Gerard Salton from early 1960s in Harvard & by Chris Buckley later in Cornell
  - Designed for laboratory experiments in IR
    - Easy to mix & match different weighting methods
    - Really terrible user interface
  - Most commonly used academic prototype
Vector Space Model: Examples

- **Lucene** (1998)
  - Originally written by Doug Cutting during 1997-1998
  - Popular open source engine written in Java

- **Elasticsearch** (2010)
  - Open source engine (based on Lucene) by Shay Banon
  - Scalable, distributed
    - Elasticsearch 5.4.1 ([https://www.elastic.co/](https://www.elastic.co/)) in 2017

- Most Web search engines are similar
Web Graph: Authorities & Hubs

This page links to many other pages (Hub)

More informative / important

Many pages link to this page (Authority)
Web Search: PageRank Algorithm

- A link analysis method used by Google [Brin & Page, 1998]
  - Named after Larry Page
  - Patented to Stanford University (not to Google)
    - Google has exclusive license rights, with Google shares (sold for $336 million in 2005)

- Ranks web pages just by authority (i.e. with in-links)
Dead End

- The web is full of dead ends
  - Random walk can get stuck in dead ends
  - Long-term visit rates are not well-defined

There is no out-link from \{2, 3, 4\}
Teleporting

- Jump to a random web page

There is no out-link from \{2, 3, 4\}
Teleporting Random Walk

- Consider the following web graph with 3 nodes
- Calculate the transition probability $P$ for the surfer’s walk with teleport probability $\alpha = 0.5$

![Diagram of a web graph with nodes 1, 2, and 3. Links are labeled with probabilities: 1/3, 1/3, 1/3; 1/3, 1/3, 1/3; 0, 1, 0; 0, 1, 0; 1/2, 0, 1/2.]

\[
\begin{align*}
\text{Link Matrix } A & \\
\begin{pmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
\end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{Original } P & \\
\begin{pmatrix}
0 & 1 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} \\
0 & 1 & 0 \\
\end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
\alpha & \times \text{Non-dead ends: teleporting} \\
\begin{pmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
(1-\alpha) & \times \text{Non-dead ends: following links} \\
\begin{pmatrix}
0 & 1 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} \\
0 & 1 & 0 \\
\end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{Final } P & \\
\begin{pmatrix}
\frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\
\frac{5}{12} & \frac{1}{6} & \frac{5}{12} \\
\frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\
\end{pmatrix}
\end{align*}
\]
Basic Algorithm: Weighting of Pages

- How to weight web pages? – using the probability vector

- Initially, all pages have equal weight of 1/6

\[ \bar{x}_0 = \left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right) \]

- Recalculate weights after one step (using the transition probability matrix \( P \))

\[ \bar{x}_1 = \bar{x}_0 P = \left(0.055, 0.208, 0.292, 0.347, 0.042, 0.055\right) \]

- Note that the calculation is done using “in-links” only
Basic Algorithm: Iteration

- Iterate: \[ \bar{x}_k = \bar{x}_{k-1}P \]

\[ \bar{x}_0 = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix} \]

\[ \bar{x}_1 = \begin{pmatrix} 0.055 & 0.208 & 0.292 & 0.347 & 0.042 & 0.055 \end{pmatrix} \]

\[ \bar{x}_2 = \begin{pmatrix} 0.013 & 0.305 & 0.465 & 0.187 & 0.013 & 0.013 \end{pmatrix} \]

... converges to...

\[ \bar{x}_n = \begin{pmatrix} 0.000 & 0.398 & 0.398 & 0.198 & 0.000 & 0.000 \end{pmatrix} = \bar{\pi} \]

- Steady-state probability = long-term visit rate = PageRank
Other Topics in IR
Text Classification

- Assigning a document to one or more classes (or categories)
  - Manually
  - Automatically

- Popular algorithms:
  - Naïve Bayes
  - \(K\text{-NN}\) (K-Nearest Neighbor)
  - SVM (Support Vector Machine)
  - Deep Learning
Text Clustering

- Cluster analysis of textual documents
  - Automatic document organization
  - Topic extraction
  - Fast information retrieval or filtering

Popular algorithms:
- K-means
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- Deep learning
Text Summarization

- Shortening a document
  - Create a summary with the major points of the original document
  - Make a coherent summary

- Popular algorithms:
  - LexRank, TextRank
  - LDA (Latent Dirichlet Allocation)
  - Deep learning
Sentiment Analysis (Opinion Mining)

- Determining the emotional attitude behind a piece of text
  - Positive, Negative or Neutral

- Algorithms:
  - Lexicon-based
  - Machine Learning (SVM)
  - Deep Learning (RNN, LSTM)
Question Answering

- Automatically answering questions posed by humans in a natural language

- Algorithms:
  - Rule-based
  - Machine Learning
  - Deep Learning
Information Extraction

- Automatically extracting structured information from unstructured (and/or semi-structured) documents

<table>
<thead>
<tr>
<th>COMPANY</th>
<th>PRODUCT</th>
<th>DATE</th>
<th>PRICE</th>
<th>REGION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung</td>
<td>Galaxy S5</td>
<td>April 11</td>
<td>?</td>
<td>U.S.</td>
</tr>
<tr>
<td>Nintendo</td>
<td>3DS</td>
<td>March 27</td>
<td>$250</td>
<td>North America</td>
</tr>
</tbody>
</table>

**PRODUCT RELEASE**

Unstructured Web Text → Structured Sequences

- Sign of the Zodiac:
  1. Aries
  2. Taurus
  3. Gemini...

- Most Common Cause of Death in America:
  1. Heart Disease
  2. Cancer
  3. Stroke...

- Largest rodent in the world:
  1. Capybara
  2. Beaver
  3. Patagonian Cavies

Strokes are the third most common cause of death in America today.

No study would be complete without mentioning the largest rodent in the world, the Capybara.
Questions?