Fundamental difference b/w IR & NLP

IR is focused much more on bag of words model (does not care about the sequence).

A is a student of B & B is a student of A is equivalent for an IR model. (But this is changing now).

We can store the tf·idf even in an evolving corpus without too much overhead because idf remains almost constant (new documents will usually have same distribution & changes are in log scale). ∴ Occasionally recalculating idf is sufficient.

Where is phonetic correction required

i) Voice search

ii) Queries containing names (& other NE thing) which sound the same but are spelt differently in different regions.

Vector space models

An example of where Euclidean distance is a bad idea:

Consider $A = D_1$ & $B = D_1 D_1$ (D_1 concatenated with D_1)

B will be usually farther to a query than A (considering query to be small); even though their contents are same.

Since we're considering bag of words model, length normalisation is fine because we anyways don't look at the context of words.

For building vector for query, it makes more sense to use tf = 1 than using tf = no of times the ~~word~~ term occurs. (cause using the same token multiple times in query should not build a bias towards terms)

Normalising the query vector is not required (because we are only concerned with the order of similarity & not normalising just scales all the similarity by the same value)

Rocchio Algorithm :

Value of q_{opt} is obtained by differentiating $(sim(q_{opt}, C_r) - sim(q_{opt}, C_{nr}))$.

For IR task : Accuracy is not a good measure of performance :

Reason : Usually, no of non relevants documents >>> no of relevants documents. Even if our system marks everything as non relevants, it will still have a very high accuracy even though the model will not be good.

4 services that IR systems provide

→ Search

→ Discovery

→ Summarisation

→ Recommendation.

Implicit feedback:

Used in a search session.

Relevance feedback: Uses documents to improve the query

Query expansion: Uses query log to improve query.

Precision can increase as recall increases. But usually that is not the case.

Recall is not easy to measure for an unannotated dataset.

Why precision @ k is not a good measure:

Consider queries that do not have large no of relevant documents (unpopular queries). Even if system retrieves all these docs, precision @ k will be very low.

NDCG

Ideally we want the documents to be presented in the decreasing order of relevance. We take this retrieval (docs in decreasing relevance order) as the ideal retrieval to calculate Z_{kj}

Link prediction problem

Given a set of nodes we have to predict which pairs of nodes have an edge b/w them.
Given a graph & a node, predict the set of nodes this node will have an edge to.

NDCG can be used to measure the accuracy of link prediction algorithm.
(Rank the links according to the probability the algorithm predicts & use the ground truth relevance to calculate NDCG).

Kappa measure for inter judge agreement

measure how much the human judges agree.
Used to ~~the eliminate human bias~~.

Kappa value ranges from $\frac{-P(E)}{1-P(E)}$ ($P(A)=0$) to 1 ($P(A)=1$).

★★ Note: $P(A)$ can be less than $P(E)$ (They're independent)

We don't have a more fast measure for marginal relevance. Usually measured fast has using click stream data & cached.

Kappa measure for inter judge agreement:

P(E) → Dependents on the density of the various outcomes & not dependents on the individual behaviours.
i.e P(E) depends on the ground truth number of relevant & non relevant documents. But because we don't have the ground truth value, we use the annotations as a proxy. If we had ground truth values, P(E) would have been:

$$\left(\frac{\#\text{ relevant docs}}{\#\text{ total docs}}\right)^2 + \left(\frac{\#\text{ irrelevant docs}}{\#\text{ total docs}}\right)^2.$$

VB encoding

Average waste = 4 bits + $\frac{4}{8} \times K$ bits

↳ last byte will be half filled on average

↳ 1 bit in every K byte

Gamma encoding

Seldom used because not byte aligned.

Language model

P(STOP) regulates the length of sentences.

Parameter estimation

$$\hat{P}(t|M_d) = \frac{tf_{t,d}}{|d|}$$

is an estimate because the document is not an actual language model but just a sample from a master distribution (i.e. document does not represent a language but a sample from the prob distribution of the language). We are just estimating master distribution from document distribution.

Which is better measure of info amount: collection frequency or document frequency?

Document frequency because there might be some documents which might contain a term large no of times irrationally increasing the collection freq.

thus

Advantage of vector space / probabilistic retrieval model over classification (learning) & generation models:

→ Vector space / probabilistic models are computationally cheap.

→ Generation models are usually closer to info need (but the information might be hallucinated).

Term doc frequency matrix: Has t.f instead of 1/0 values.

Time complexity of

OR/ AND → $O(|x|+|y|)$

NOT (just NOT) → $O(N)$

x AND NOT y → $O(|x|+|y|)$

x OR NOT y → $O(N)$

F score

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

See soundex encoding

Kappa measure

$$= \frac{P(A) - P(E)}{1 - P(E)}$$

Probabilistic IR

$$P(R=1|d,q) = \frac{P(\vec{x}|R=1,q)\,P(R=1|q)}{P(\vec{x}|q)}$$

$$\therefore O(R|d,q) = \frac{P(\vec{x}|R=1,q)\,P(R=1|q)}{P(\vec{x}|R=0,q)\,P(R=0|q)}$$

$$= O(R|q)\,\frac{P(\vec{x}|R=1,q)}{P(\vec{x}|R=0,q)}$$

separate into term product, sub $u_t = p_t = 0.5$ for terms not in query, you get:

$$O(R|d,q) = O(R|q) \prod_{t \in x_t = q_t = 1} \frac{p_t}{1-p_t}\,\frac{1-u_t}{u_t}$$

$$RSV_d = \sum_{t: x_t = q_t = 1} \log \frac{p_t}{1-p_t} + \log \frac{1}{1/u_t} \log\left(\frac{1-u_t}{u_t}\right)$$

Estimate $u_t = \frac{df_t}{N}$, p_t = constant = 0.5
(can also be estimated as no of docs relevant docs that have t)

$$RSV_d = \sum_{t: x_t = q_t = 1} \log \frac{N}{df_t}$$

Okapi / BM25

$$RSV_d = \sum_{t:\, q_t = d_t = 1} \log \frac{N}{df_t} \cdot \frac{(k_1+1)\, tf_{t,d}}{k_1\left((1-b) + b \times \frac{L_d}{L_{avg}}\right) + tf_{t,d}}$$

Language models for IR.

$$P(t|d) = \lambda\, P(t|M_d) + (1-\lambda)\, P(t|M_c).$$

longer queries → prefer lower λ
shorter queries → prefer larger λ.

NDCG

$$= Z_{KG} \sum_{i=1}^{k} \frac{2^{R(i,m)} - 1}{\log_2(1+i)}$$

Naive Bayes assumption in IR:

Positional independence: Probability of particular token to occurring in document d given class c is independent of its position.

Although usually violated, does not matter much because the order usually remains same (order of docs)

Usually: Micro average has higher value than macro average:

Reason: Precision is usually low for the tail classes. In macro macro average, all classes receive the same weight (& tail classes are usually high in number)

Mutual importance:

For the first time we look at terms other than the query terms that are important to answer the query

Applications of retrieval systems:

Summarization

For a document of length L, a summary of $\frac{L}{2}$ length was considered acceptable, but with increase of docs on web, we want summaries to be orders of magnitude less than L. (Nothing greater ~~than~~ than $\frac{L}{10}$ is considered acceptable.

Types of summaries

i) Indicative v/s informative

Like headlines — Has some content drawn from the main doc
Very precise & short

ii) Extractive v/s abstractive (generative)

Extracts parts from the main doc to build the summary — Rewrites the info in the doc in its own "supposedly coherent" version.

iii) Generic v/s query oriented

Provides the author's view — It provides the summary as per the user query / demand.

iv) Backgrounds v/s just the news

(Most relevant to news articles)

Assumes that the reader has no prior knowledge — Assumes that the reader is well aware of the situation

v) Single v/s multi documents

Summary from single doc — Multiple docs are collated to generate summary.

Some important parts of documents that help in summarization

→ Abstracts & conclusions
→ Section headings in scientific papers.

→ Many such meta linguistic cues hidden in general documents too. Ex: Phrases like "in summary" are direct sources of summary.

→ Font differences (bold, italics, larger font size) are usually cues to important info.

Graph based summary algorithms

→ Build a graph with nodes & edges that somehow represents the document.
→ Measure certain properties of this graph to build the summary.

→ The algorithm anchors upon the concept of "Prestige" in social networks.

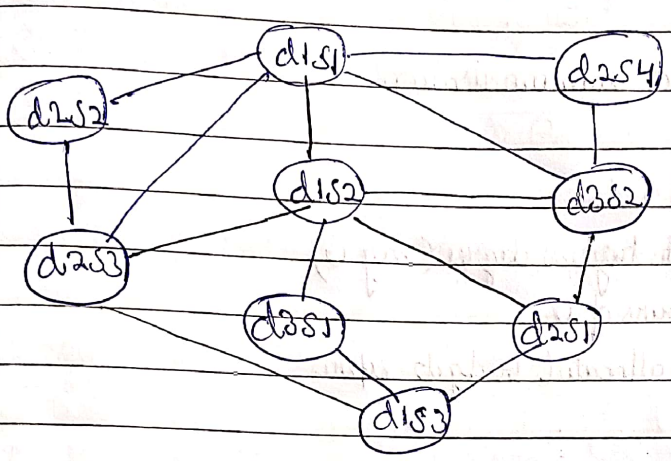
→ Imagine you're a graph where: nodes are sentences & edges are similarity b/w sentence pairs.

→ Nodes constructed using tf-idf vectors of sentences.
→ Edges constructed using cosine similarity b/w the tf-idf vectors.

Each sentence represented with an ID : dXSY
doc X, sentence Y.

d2S3 : 3rd sentence of 2nd doc.

Consider the graph



— = <0.5
— = ≥0.5.

In the above graph, it is apparent that d1S2 must be present in the summary ∵ cause : high degree : similar to most other sentences & most of its edges have high weight ∴ This idea is called degree based summarization approach.

Noisy links :

Links that cause spurious similarities. Must be removed from the graph.

Removed by setting a threshold T to on similarity score below which we'll remove the edges.

High T → extremely sparse graph
low T → too much redundant info (noise)
Value of T chosen on validation set.

Binarization of graph

After removal of edges below certain threshold, forget the weights (convert weighted graph to unweighted graph).

Degree centrality based summarisation

① Compute degree of each node
② Pick the node with highest degree (say v)
③ Remove all neighbours of v
④ Repeat ① till the allocated budget expires.

Pagerank

Recursive idea of recursive importance : a node's importance is determined by how important its neighbours are.
The graph remains same as before but rather than computing degree, run pagerank algorithm.

① ~~Run pagerank~~
②

2 possible algorithms:

I) ① Compute pagerank of nodes
② Select node with highest pagerank
③ Remove neighbours
④ Go to ②

II) ① Compute pagerank of nodes
② Select node with highest pagerank
③ Remove neighbours
④ Recompute pagerank, go to ②.

Called the textrank / lexrank ~~algorith~~ algorithm.

linguistic summarisation algorithms

Based on the idea of lexical chains.

lexical chains summarisation:
collecting information from correlated chain of words

Build chains:
ⓐ Pick a set of candidate words (generally, the words are nouns / noun compounds)
ⓑ For each candidate word, build a chain based on its relatedness with other words.
ⓒ If the candidate word is already related to words in a particular chain, then just it with that chain rather than building a new chain for it.

Measure of relatedness

Usually rely on lexical resources like Wordnet.

Choosing one of the chains as a document representative

Choose the strong chains
↳ Determined by a scoring mechanisms.

Heuristics for scoring:

i) length: No of occurences of a particular words of the chain.

ii) Homogenity Index:
$= 1 - \frac{(\text{No of distinct occurences of a words of a chain})}{\text{length}}$

Score (chain): length × Homogenity index.

Ex:
word #of occurences.
Chain: Microsoft (10) Concern (1) Company (6) entertainment (1)
Enterprise (1) MIT (1)

length = 20 (10+1+6+1+1+1)

$HI = 1 - \frac{6}{20} = \frac{14}{20}$

Score = 14.

Strong chain: Chains with score higher than Average (score) + 2 X standard deviation (score).

If there are 3 strong chains (say 1, 2 & 3): Use these strong chains to build summary.

Possible summary strategies:

i) Choose the sentence that has the first occurrence of word from the chain (i.e the first sentence that contains any word from sentence). Keep doing till budget runs out.

ii) Choose the first sentence that contains the occurrence of the key word (candidate word that started the chain) of the chain.

iii) Choose the sentence that has the first occurrence of a high density (choose a threshold) of words from the chain.

For practical purposes, ii) method works best (because it's works based on the key-word & other words in the chain are related to this word)

Wordnet based summarization

Idea:

(1) Select sentences based on their semantic contents.
↳ meaning.

(2) The contents is given relative importance w.r.t the semantics of the whole text.

~~(3) Reduce the text~~

(3) Reduce the pieces of text corresponding to the same semantic contents : Reduction of redundancy.

Key steps:

i) Preprocessing
ii) Subgraph construction from wordnet.
iii) Synset ranking
iv) Sentence selection
v) PCA.
vi) Final framing.

Preprocessing:

a) Split the text into sentences (imp. for languages like Mandarin)
b) POS tagging (every word in the sentence is tagged with its most relevant POS) : Helps to deduce the correct sense of the word.

c) Identifying collocations → words that typically appear together in a sentence.

Ex: all idiomatic phrases.

d) Remove stopwords

e) Note: The sequence of the preprocessing steps is very important. (Due to phrases like "take off")

Subgraph construction:

a) Mark all the words & collocations that appear in the text to be summarized in the wordnet

b) Traverse the generalisation edges upto a fixed depth & mark the synsets you visit. → hyponymy edges of wordnets.

↳ groupings of synonymous words that express the same concepts (parts of wordnet)

Ex: book.n.01 → [book.n.01, collocation.n.02, impression.n.06, magazine.n.01 ...]

noun (diff senses of word) — has synset

Each node also has a gloss (the meaning) & a synset.

c) Construct a graph containing only the marked synsets as nodes & the generalisation relationships as edges → synset subgraph.

Synset ranking : Rank the synsets based on their relevance in the text (to be summarized)

Steps

i) Construct a rank vector R corresponding to each node of the graph. (R has 1 entry for each node)

Each entry $= \frac{1}{\sqrt{n}}$ n = no of nodes/synsets in graph.

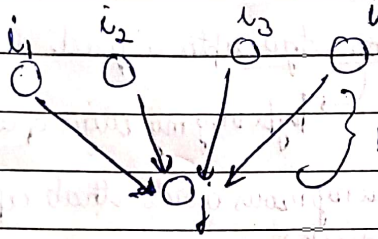
ii) Authority matrix

$A(i,j) = \frac{1}{\text{num of predecessor}(j)}$ if i is child of j

$= 0$ otherwise

predecessor in wordnet

Ex.



Authority matrix determines how many nodes does j derive its meaning from.

iii) Update rank vector

$$R_{new} = \frac{R_{old} * A}{|R_{old} * A|}$$

still R_{new} changes less than a predefined threshold.

Each time we compute R_{new}, we add a small bias to all its entries to prevent underflow.

Sentence selection

(a) Construct a matrix M with m rows & n columns (m → no of sentences in the text to be summarised; n → no of nodes in subgraph)

(b) For each sentence S_i, again traverse the subgraphs following the generalisation edges with the words present in S_i.
Find all the synsets that are reachable. SY_i
For each $sy_j \in SY_i$, set $M[S_i][sy_j] = R[sy_j]$

PCA

→ Compute the principal components of M.
↳ Eigen vectors of M with the largest eigen values.
These eigen vectors will capture the important meaning directions.

→ Sort the eigen vectors based on eigen values. Take the top few eigen vectors & compute projection on each sentence.

$$Pr(\vec{e})_{\vec{s_i}} = \frac{\vec{e} \cdot \vec{s_i}}{|\vec{s_i}|}$$

→ How many sentences to project on?
Project each eigen vector on K sentences, where

$$K \propto \frac{\lambda_i}{\sum_j \lambda_j}$$

$$K = \left(\frac{\lambda_i}{\sum_j \lambda_j}\right) N \longrightarrow N = \text{no of sentences in target budget.}$$

ie project on all sentences & choose the top k sentences for each eigen vector.

~~6.~~ 7.

Final pruning

Removal of undesired references:

@ Remove sentences that start with pronouns.

(b) Remove sentences within quotes.

↳ Because usually represent opinions & not facts.

Optimization based summarization

→ A global inferencing method.

→ Notation:

D is a document

It has t_n textual units (sentences in this case)

D: $t_1, t_2 \cdots t_n$

→ Rel(i): The relevance of t_i to the target summary.

→ Red(i,j): Redundancy b/w t_i & t_j

→ l(i): length of text unit t_i (in terms of no of words)

→ The inferencing is modeled as follows:

- Problem is to select a subset of S of the ~~document~~ textual units from D such that the summary score S(S) is maximised.

$$S(S) = \sum_{t_i \in S} Rel(i) - \sum_{t_i, t_j \in S, i \neq j} Red(i,j)$$

$$\therefore \text{Summary} = \arg\max_{S \subseteq D} \left[\sum_{t_i \in S} Rel(i) - \sum_{t_i, t_j \in S, i \neq j} Red(i,j) \right]$$

↑ Increase the relevance of text units in S.

↕ reduce the redundancy among the choice of text units.

→ $Rel(i)$, $Red(i,j)$ → should be something that enforces diversity.

↑ would be something like presence of special entities

Solving this optimisation problem:

i) Greedy solution

→ Sort D such that $Rel(i) > Rel(i+1)$ $\forall i$.

→ $S = \{t_1\}$ → most relevant to text units.

→ while $\sum_{t_i \in S} l(i) \leq K$ ← budget

$$t_j = \arg\max_{t_j \in D - S} s(S \cup \{t_j\})$$

$$S = S \cup \{t_j\}$$

return S

★ Greedy solution provides approximate results. Ale much better solution

* We'll restate this optimisation problem as an Integer Linear Programming problem
↓
has many solving methods

Recasting this problem as constrained optimisation function.

Syntax used ILP formulation:

Optimisation function

maximise $\sum_i \alpha_i Rel(i) - \sum_{i,j} \alpha_{ij} \cdot Red(i,j)$

constraints : $(\forall i, \forall j)$

$\alpha_i, \alpha_{ij} \in \{0, 1\}$ ← Indicator variables : tells us whether a textual unit or textual pair are indicated in the target summary or not

$\sum_i \alpha_i l(i) < K$ ← budget constraint

$\alpha_{ij} - \alpha_i \leq 0$

$\alpha_{ij} - \alpha_j \leq 0$ ← If b_i & b_j pair is included then they must be individually included.

$\alpha_i + \alpha_j - \alpha_{ij} \leq 1$ ← If b_i & b_j are individually included then the pair of b_i & b_j must be included

How to operationalise Rel(i) & Red(i,j):

Depends on a lot of things such as:

i) Domain of the summarisation

Ex: For news summarisation, we could use

$Rel(i) = POS(t_i, D)^{-1} + SIM(t_i, \mathcal{D})$

where $\mathcal{D}$ = documents collection

D = documents being summarised

$POS(t_i, D)$ = position of textual unit t_i in documents D.

$SIM(t_i, \mathcal{D})$ = similarity of text unit with overall collection of documents.

Empirical observation: initial sentences in a news document is very important.

$SIM(t_i, \mathcal{D})$: $\overrightarrow{t_i}$ — vectorisation of ~~document~~ t_i

$\overrightarrow{\mathcal{D}}$ → vectorisation of $\mathcal{D}$ → one way is to sum up the vectors of all individual docs.

$SIM(t_i, \mathcal{D}) = \text{cosine sim}(\overrightarrow{t_i}, \overrightarrow{\mathcal{D}})$

Ex: ~~Query focused summarisation:~~

~~Given query Q we want~~

III) $Red(i,j) = SIM(t_i, t_j) = \text{cosine sim}(\overrightarrow{t_i}, \overrightarrow{t_j})$

b) Q 8c: Query focused summarization:

* Given a query Q → we want a relevant document summary.

Here position of t_j in doc is not much imp but presence of query terms in t_i is imp.

$$Rel(i) = SIM(t_i, Q) + SIM(t_i, D)$$

$$Red(i,j) = SIM(t_i, t_j)$$

Alternate problem formulation: concept based ILP:

No longer at text units or sentence level but at the concept level.

Tells abt the important concepts that the summary should cover.

① Summary benefits by including a particular concept only once

② This means redundancy is implicitly captured

Let c_i ← indicator variable for the concept i: (present or not)

w_i ← weight of this concept.

Revised ILP

maximise $\sum_i w_i c_i$

subject to: $\sum_j l_j s_j \leq L K$ — s_j = indicators of sentence j is summary

$s_j occ_{ij} \leq c_i \quad \forall i,j$ — occ_{ij} = occurrence of concept i in sent j.

↳ If you pick a sentence: it mandates all concepts in that sentence should be chosen.

$\sum_j s_j a_{ij} \geq c_i$

↳ If a concept is ever selected, its mandate that atleast 1 sentence containing that concept must be selected.

$c_i \in \{0,1\} \quad \forall i$

$s_j \in \{0,1\} \quad \forall j.$

& One simple definition of concept: "word bigrams" & weight = no of occurrences of word bigrams in the input documents.

Evaluating summaries

i) ROUGE score: (Recall oriented understudy of Gisting Evaluation)

→ Recall is important in summarisation because we are more interested in "coverage" than precision (because we want to cover more in limited budget).

Precision is more important in safety critical system.

a) ROUGE-N (N-gram) (N = 1, 2 ...)

→ Given a machine generated summary (candidate summary): C & a reference summary (human written summary / gold standard summary): R.

Ex: R: the cat is on the mat

C: the cat is and the dog

Rouge-N precision	Rouge N recall (more imp)
Ratio of the number of N grams in C that also appear in R by the number of N grams in C	Ratio of number of N grams in C that also appears in R over no of N grams in R.

For the given example:

ROUGE-1 =

precision = $\frac{3}{5}$ intersection unigrams: {the, cat, the}
unigrams in C: {The, cat, and, the, dog}

recall = $\frac{3}{6}$ unigrams in R: {the, cat, is, on, the, mat}

Rouge 2 precision = $\frac{1}{4}$ bigrams in C: {the cat, cat and, and the, the dog}

Rouge 2 recall = $\frac{1}{5}$ R: {the cat, cat is, is on, on the, the mat}

Rouge N F1 = $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

b) ROUGE-L : longest common subsequence

longest common sequence → not necessarily consecutive.

LCS = the cat the

|LCS| = 3

Denominator = no of unigrams of C (for precision) & R (for recall)

Rouge L precision = $\frac{3}{5}$

Rouge L recall = $\frac{3}{6}$.

If R & C contain multiple sentences: compute value of each sentence pair of R & C & average.

c) Rouge-S (skip grams)

R: The cat is on the mat.
C: The gray cat & the and the dog.

Skip = S: we can skip S unigrams & if the n grams still match, we count it as match.

Eg: For the above example: S=0 → 0 match
S=1 → 1 match.

Suppose we have generated the ILP summary: i.e we have selected some sentences.

Further issues:

i) Ordering of selected sentences

Domain dependent.

Eg:- In news summarisation: order in the same order as they appear in doc.

→ If coherence is imp: Choose ordering that makes neighbouring sentences similar (cosine)

→ In multi entity summary (summary that talks abt 2 or more things): Choose ordering that brings similar entities closer.

→ Topicality: If summary covers multiple topics, make the sentences coherent as well as topically coherent.

ii) Simplifying sentences:

Parse the output summary & decide based on some rules which parts to clip off.

Some basic rules that can be followed:

i) Initial adverbials: Ex: "On the other hand", "as a matter of fact" etc.

ii) Prepositional phrases without named entities:

Ex: The commercial fishing restrictions in Washington will not be lifted until salmon population increases to a substantial number. X

iii) Attribution clauses:

attributes the sentence to the sayer.

Ex: Rebels agreed to talks with officials ~~said observers said on Tuesday~~ X

iv) Appositives: parts in b/w commas

X

Rajan, 28, an artist who was at that time living in Philly, ~~found~~ found inspiration in Bath.

Information extraction

① We want to obtain structured text from unstructured text

Objectives of an IE system:

→ Find & understand limited relevant parts of text

→ Gather info from many different texts.

→ Produce a structured representation of the relevant info gathered (may be a rdbms, knowledge base etc)

End goals:

→ Make the data "useful" for users

→ Organize the info in a semantically precise manner so that it can be used by algorithms (computers)

Input: unstructured / semi structured data $\xrightarrow{IE}$ structured data.

→ Concerned with exact & factual information. We do not want opinions.

Roughly: An IE system would want to answer:

Who did what to whom & when.

Ex: In 1998, Larry Page & Sergei Brin founded Google Inc.

Info: Founders (Google Inc, Larry Page)

Founders (Google Inc, Sergei Brin)

Founded In (Google Inc, 1998)

relation Entity1 Entity2

Applications

Biomedical domain:

→ Starts with large no of scientific publications

→ We want to have a table lookup that allows us to discover easily innovations related to i) particular genes ii) particular proteins etc.

→ Biomed entities have many ambiguous names & lots of synonymous terms.

→ We want to automatically identify the biomedical entities & link them (store as a graph) in knowledge base.

We want to convert the running texts into tables like

subject	relation	object
p53	is a	protein
p53	has function	apoptosis
apoptosis	involved in	cell death

News articles:

Relations here are slightly more well defined.

Relations	Examples	Types
affiliation		
→ personal	Married to, mother of	PER → PER
→ organisational	president of, spokesman for	PER → ORG
→ Artifactual	owns, invented	PER/ORG → ART
Geospatial		
→ proximity	near to	LOC → LOC
→ directional	south of	LOC → LOC

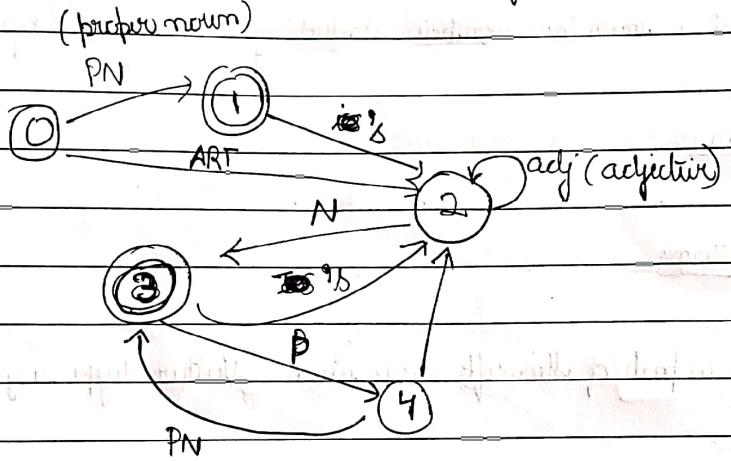
Bank of → Organisations → political
a unit of / parent of annexed / acquired
ORG → ORG
GPE → GPE
↓
geopolitical entity

Some trivial methods

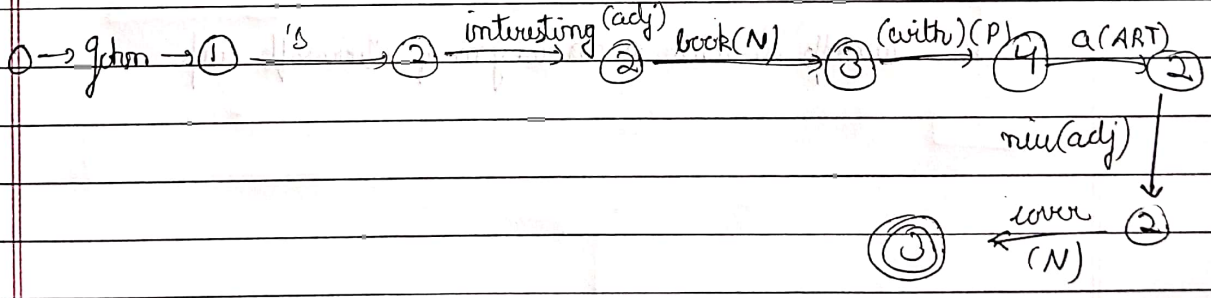
i) Handwritten rules (regex based)
ii) Bootstrapping methods
iii) Supervised methods.
iv) Distant supervision
v) Unsupervised.

REGEX based IE

Use finite automaton for noun groups :



Ex: John's interesting book with a nice cover.



Many such regex to extract info:

Ex: To extract which person holds what position in what organisation, we could use regex like

[person], [position] of [org]

Ex: Braskovic, leader of the Serbian Renewal movement -

or [ORG] (appointed, named etc) [person] prep [office].

Hyponymy as a key to IE (specialised regex)

Rules: (some representative rules)

Y such as X ((,X)* (, and/or) X)

such Y as X : such authors as Shakespeare

X or other Y: bruises, wounds or other injuries

X & other Y : temples, monastries & other civic buildings

Y including X : common law countries including Canada & England

Y, especially X

X ← hyponym

Also called Hearst patterns

Microsoft research is a part of Microsoft corporation: Neither hyperonymy nor hyponymy.

These are called Meronymy & Holonymy.

Microsoft research is meronym of Microsoft corp

Meronymy, holonymy based Hearst patterns

Ex: ~~part~~ [NN-PL] of [PREP] {the|a} [DET] [JJ|NN]* ~~whole~~ [NN].

parts → [NN-PL]
whole → [NN]

Ex: basements of a building

Ex: whole('s) [POS-variety] : part
[NN]/[NN-PL] [NN-PL]

Think which POS's can fit.

Disadvantages of Hearst patterns

→ Hard to write rules for every ~~sit~~ situation
→ Hard to maintain rules.
→ Explodes very fast
→ Changes with domain of documents (news / biomedical etc)

Generally lead to only an accuracy of 66% on Hyponymy & abt 55% on meronymy (not acceptable)

B

Bootstrapping approach

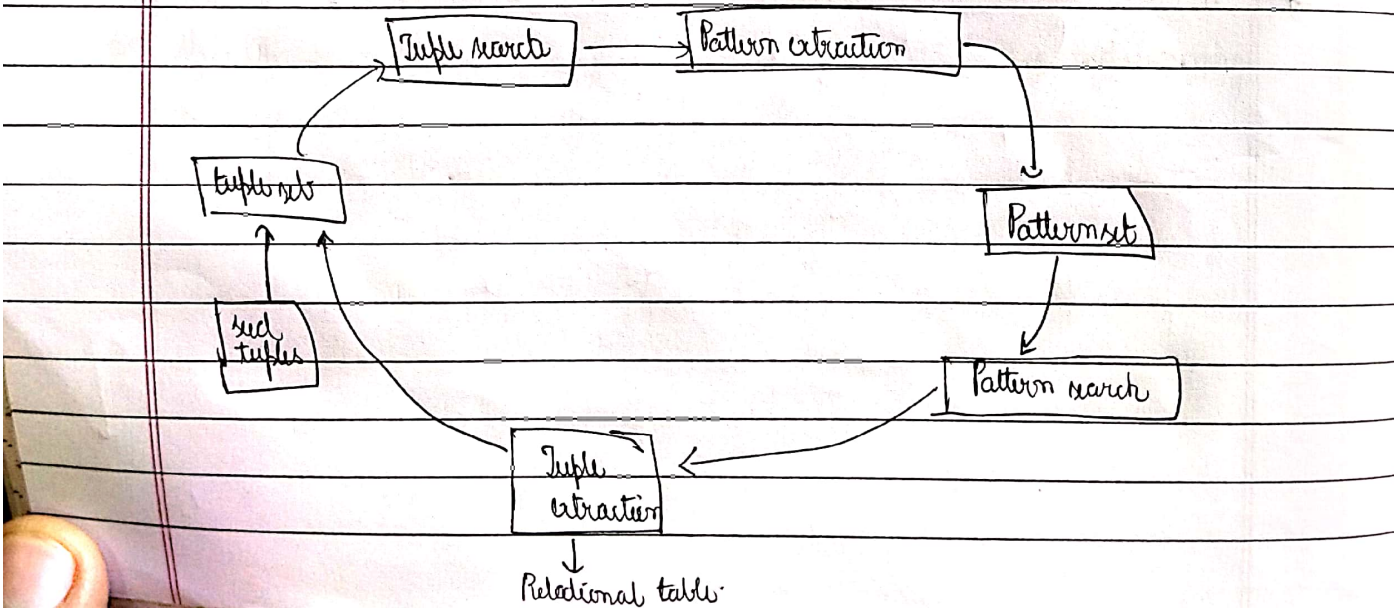
↳ we need something to bootstrap on.

→ Not too much data is available for full fledged supervised training.
→ But we have a small seed set of annotated data.
→ We follow semi supervised approach.

Ex: Target relation: Burial place
Seed set: [Mark Twain, Elmira]
→ Expanding the knowledge using seed set:
Google search "Mark Twain" & "Elmira".
We get results like "Mark Twain is buried in Elmira"

"The grave of Mark Twain is in Elmira" & "Elmira is Mark Twain's final resting place"

→ Build Hearst patterns using the obtained results. Ex: X is Y's final resting place.
→ search google for these patterns to obtain more tuples to expand the seed set & repeat.



Disadvantages

→ Extremely sensitive to the seed set.
→ A large number of hyperparameters (when to stop, how many search results to take, how many patterns to add)
→ Probabilistic notion is missing (Not sure how confident we are with the results).

Supervised relation (information) Extraction

→ Choose a set of relations that you want to discover.
→ Find & label data. (Expensive)
  ★ Choose a representative corpus
  ★ ~~Hand~~ Label entities
  ★ Label relations (Expensive because needs expert intervention)
→ Divide into train, dev & test
→ Train a supervised ML model & predict ~~the~~ on the test set.

We model supervised IE as a classification task
We break it into a two step process.
① Given a pair of entities, we predict if they are related or not
② If a pair is related, predict the relation type.
  ↳ multi label classifier / multi class classifier

Feature Engineering for supervised learning.

→ Consider the entity as bag of words
Consider the example: American Airlines, a unit of AMR, followed with the same move, spokesperson Tim Wagner said.

→ Feature 1: Bag of words
WM1: {American, Airlines}.
WM2: {Tim, Wagner}.
→ Feature 2: Head words.
HM1: ~~Am~~ Airlines HM2: Wagner, HM12: Airlines + Wagner
→ Feature 3: ~~W~~ left context / right context
M2-1: spokesperson M2+1: said.
→ Feature 4: Span of bag of words b/w the entities (models long range dependency)
{a, AMR, of, unit, followed, with, move, same, spokesperson}.
→ Feature 5: Category of the named entity (gives type of the relation)
M1 → org M2 → PER.
M12 → ORG - PER.
→ Feature 6: Name / Nominal / Pronoun.
M1: EL → name
M2: EL → name
Needed to correctly link the pronouns to their named entities

→ Feature 7: Dependency parse (see slides for tree)
Can use the words in the dependency tree (gives dependency info)
Ex: HIDW1: ~~was~~ followed: airlines
HIDW2: said: Wagner.
also: path: {airlines, followed, said, Wagner}.

→ Feature 8 : Constituency parse features (see slides for this)

→ Feature 9: Trigger words which are possible sources of relationship.
Ex: parents, father, wife, husband
Can be recognised using wordnet & some gazzeters like country name list for geopolitical relations.

Use these features on standards ML models like SVM.