Part III: Semantics

Notes on Natural Language Processing

Chia-Ping Chen

Department of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan ROC

Introduction

- **Semantics** is the *meaning* of sentences.
- A sentence consists of phrases, which consists of words, which consists of morphemes. There is a syntactic hierarchy.
- Is there a semantical hierarchy as well? How is the meaning of a sentence related to the meanings of the phrases, words and morphemes?
- Computational semantics is used in practical problems such as question answering and information extraction.

Semantic Analysis

- **Semantic analysis** takes a sentence (input) and constructs a meaning representation (output).
- Common methods of meaning representation
  - first order predicate calculus
  - semantic network
  - conceptual dependency
  - frame-based representation
- For example, the representations for

  I have a car

  are illustrated in Figure 14.1.

Meaning Representation

- A meaning representation scheme uses a set of symbols to represent **objects** and **relations** between objects.
- In this example, the speaker, the car, and the possession are represented.
- Two perspectives of meaning representation
  - as a representation of input sentence
  - as a representation of the state of a world

  Such a dual perspective links a sentence to knowledge about the world.

Literal Meaning

- The simplest approximation to the actual meaning is the *literal meaning*.
  - That is, the *context* where a sentence occurs is not taken into account.
  - In particular, the literal meaning is different from *idioms* or *metaphors*.
- Literal meaning is simple in the sense that it is directly related to the meaning of words in the sentence.

Desiderata for Representation

- Imagine we have an automated system that accepts spoken language queries from tourists looking for help.
- The core of this system is a representation of the world. The desired properties of this representation are
  - verifiability
  - unambiguousness
  - canonical form
  - inference
  - expressiveness
- We will consider the task of giving advice about restaurants to tourists to illustrate the main points.

Verifiability

- It should be possible to compare (or match) the meaning of a sentence against the knowledge base.
- As an example, consider the query sentence:

  Does Maharani serve vegetarian food?

  which can be represented by

  Serves(Maharani, VegetarianFood).

- The system matches this representation against a knowledge base about restaurants.

Unambiguousness

- The process from sentence to representation is

  sentence $\rightarrow$ meaning $\rightarrow$ representation

- Once the meaning is determined, the representation should allow no ambiguity.
- A sentence may have several legitimate meanings, e.g.,

  I want to eat someplace that's close to ICSI.

  A good system should have the ability to tell which is likely and which is not.

Canonical Form

- It is desired that sentences with the same meaning should be assigned the same representation.
- This is the notion of **canonical form**.
- Using canonical form simplifies reasoning/matching tasks.
- However, the task of semantic analysis becomes more complicated. Need to deal with different words and syntactic structures.

Inference and Variable

- **Inference** refers to a system's ability to draw valid conclusions based on the meaning representation of input and/or its store of knowledge.
- Some queries may require facts about the "world".

Can vegetarians eat at Maharani?

- A **variable** is used in the representation of a query which does not refer to a specific entity.

I'd like find a place where I can get vegetarian food.

Serves(x, VegetarianFood)

Expressiveness

- The **expressiveness** of a meaning representation scheme is a measure of the various meanings it can describe.
- In principle, there is a very wide range of input and knowledge base. We want a meaning representation method that can accurately represent any semantic sentences.

Predicate-Argument Structures

- The **predicate-argument structure** is a structure for meaning representation. It asserts specific relationships among the constituents of the structure.
- The meaning of the sentence

  I want Italian food. (NP want NP.)

  can be represented by a predicate-argument structure

  Want(wanter, wantedThing)

  - There are two NP arguments to this predicate.
  - The first (pre-verbal) argument is the subject.
  - The second (post-verbal) argument is the object.

Semantic Roles and Restrictions

- In the previous example, the first argument assumes the role of doing the wanting, while the second assumes the role of something wanted.
- We can associate the surface arguments of a verb with a set of **semantic roles**.
- The study of roles for specific verbs or classes of verbs is called **thematic role (or case role) analysis**.
- Only certain kinds of categories can play the role of a "wanter". (Not everything can want something.) This is called **semantic restriction** or **selectional restriction**.

Other Predicates

- Verbs are not the only objects that can carry a predicate-argument structure.
  - A preposition (under) can do.

    a restaurant under 50 *dollars*

  - A noun (reservation) can do, too.

    a reservation for a table for two persons

First-Order Predicate Calculus

- **FOPC** is a meaning representation language.
- It provides a computational basis for verifiability, inference and expressiveness.
- It makes little assumptions about how things ought be be represented.

Elements of FOPC

- A **term** is used to represent objects. There are three ways to represent a term.
  - A **constant** represents a specific object, e.g. *Henry, Maharani*
  - A **function** represents a concept associated with an object.
  - A **variable** can represent an unknown object or all objects of a kind, depending on the quantifier.
- A **predicate** is used to represent relations between objects.
- A predicate serves as an atomic formula for meaning representation. Composite representation can be formed via logical connectives.

Syntax of FOPC

Formula $\rightarrow$ *AtomicFormula* $|$ *Formula Connective Formula* $| \ \ldots$

AtomicFormula $\rightarrow$ *Predicate(Term,$\ldots$)*

Term $\rightarrow$ *Function(Term,$\ldots$)* $|$ *Constant* $|$ *Variable*

Connective $\rightarrow \ \wedge \ | \ \ldots$

Qualifier $\rightarrow \ \exists \ | \ \forall$

Predicate $\rightarrow$ *Servers* $|$ *Near* $| \ \ldots$

Function $\rightarrow$ *LocationOf* $| \ \ldots$

Semantics of FOPC

- A sentence is represented as a formula in FOPC language.

  Ay Caramba is near ICSI.

  Near(LocationOf(Ay Caramba), LocationOf(ICSI))

- It can be given a value of true/false based on whether it is true in the FOPC representation of the world.
- The truth value of a formula is determined by the truth table.

Variables and Quantifiers

- Variables are used in two different ways in FOPC.
  - to refer to particular anonymous objects
  - to refer to all objects of a kind
- These uses are made possible through the use of quantifiers $\exists$ (there exists) and $\forall$ (for all).
- Suppose we want to find *a restaurant that serves Mexican food near ICSI*. This is the same as asking the truth value of

$$\exists x\ Restaurant(x) \wedge Serves(x, Mex) \wedge Near(x, ICSI)$$

Inference

- An important method for inference is **modus ponens**

$$(\alpha \wedge \alpha \Rightarrow \beta) \Rightarrow \beta$$

- An FOPC example for modus ponens.

VegRest(Rudys) $\wedge$ *(* $\forall x$ *VegRest(*x*)* $\Rightarrow$ *Serve(*x*, VegFood)*

$\Rightarrow$*Serves(Rudys, VegFood)*

Modus Ponens

- **forward chaining**: all applicable implication rules are applied and the results are stored until no further application of rules is possible.
- **backward chaining**: start with the query we look for implication rules $\alpha \Rightarrow \beta$ with the query as β and check if α is true.

Representing Categories

- Create a unary predicate for each category of interest. For example,

  VegRest(Maharani)

- This method treats categories as relations rather than as objects.
- Alternatively, one can treat category as an object via *is-a* relation

  ISA(Maharani,VegRest).

- Hierarchy of categories can be represented by *a-kind-of* relation

  AKO(VegRest, Rest).

Representing Events

- Consider the example,

  I eat [a turkey sandwich [for lunch]] [at my desk].

- We may use a predicate for each *eat* but that is messy.
- It makes sense to say that the above examples refer to the same predicate with some of the argument missing.

  Eating(eater, eatenThing, meal, place).

- Alternatively, this can be represented quite flexibly by

  $\exists$ *w ISA(w,Eating)* $\wedge$ *Eater(w,eater)* $\wedge$ *Eaten(w,eaten)* $\ldots$

Representing Time

- In order to represent time, we need to know how sentences convey temporal information.
- We want to distinguish past, present and future. In addition, we want to know which precedes which (the temporal order).
- The tense of a verb indicates the relationship between three times: time of event, reference time and the time of utterance.

Temporal Information

- Without representing temporal information, the following examples have the same representation

  I arrived in New York. I am arriving in NY. I will arrive in NY.

  $\Rightarrow \exists\, w\; ISA(w,Arriving) \wedge Arriver(w,speaker) \wedge Dest(w,NY)$

- The temporal information should not be ignored. For example, the representation for the first becomes

  $\exists\, i,e,w\; ISA(w,Arriving) \wedge Arriver(w,speaker) \wedge Dest(w,NY)$

  $\wedge\; IntervalOf(w,i) \wedge EndPoint(i,e) \wedge Precedes(e,Now)$

  with temporal information incorporated.

Aspect

- Aspect refers to some notions related to an event, say E
  - whether E has ended or is on-going
  - whether E happens at a point in time or over some period
  - whether or not a state in the world comes about because of E
- Expression of events are divided into 4 classes
  - Stative: I know my departure gate.
  - Activity: John is flying.
  - Accomplishment: Sally booked her flight.
  - Achievement: She found her gate.

Stative Expression

- Stative expressions represent a notion that the event participant is in a certain state at a particular point of time. For example,

  I need a ticket to Taipei.

- Stative verbs, such as *like, need, have, want*, would be odd if used in the progressive form. They are also strange when used as imperatives.

  I am needing a ticket to Taipei. Need a ticket to Taipei.

Activity Expression

- Activity expressions describe events that have no particular end time. These events are seen as occurring over some span of time.
- Unlike stative expressions, it is not a problem to have progressive for activity expressions. For example

I live in a flat. I am living in a flat.

- Since the event goes for a period of time, it would be odd to use *in* with a temporal expression. *For* is OK.

I live in a flat in a month. I live in a flat for a month.

Accomplishment Expression

- Accomplishment expressions describe events that have a natural end point and result in a particular state. For example

  He booked me a reservation.

- Implicitly, there is an event occurring over some period of time that ends when the intended state is accomplished.
- One can use *stop* to test whether an expression is activity or accomplishment.

  He stopped booking me a reservation.

  He stopped living in a flat.

Achievement Expression

- (like accomplishment) Achievement expressions indicate the occurrence of an event which results in a state.
- (unlike accomplishment) The event is considered happening in an instant rather than a period of time.
- Events related to verbs *find, reach, discover* are considered to happen at once.
- One cannot *stop* an achievement event. For example,

  She stopped finding the gate.

Representing Belief

- Consider the example

  I believe that Mary ate British Food.

- The following representation

  $\exists u, v$ *BelievedProp*(u, v) $\wedge$ *ISA*(u,*Believing*)
  $\wedge$ *Believer*(u,*Speaker*) $\wedge$ *ISA*(v,*Eating*)
  $\wedge$ *Eater*(v,*Mary*) $\wedge$ *Eaten*(v,*BritishFood*)

  would be problematic, as it implies that Mary ate British Food, but belief is not fact.

Operator

- The following is an invalid FOPC,

  Believing(Speaker, Eating(Mary, BritishFood)).

  The second argument is not a term.

- In representing beliefs, we need to extend it to hold between objects and relations. This is achieved by augmenting FOPC with operators. An operator such as *Believe* takes two arguments, one for believer and the other for believed proposition.

  Believe(Speaker, $\exists v$ *ISA(*v*,Eating)* $\wedge$ *Eater(*v*,Mary)* $\wedge \ \ldots$*)*

Syntax-driven Approach

- A basic approach for semantic analysis, called syntax-driven semantic analysis, is based solely on lexicon and grammar.
- The meaning is the literal meaning, which is both context independent and inference free. There are some applications where such a restricted view is adequate.

Principle of Compositionality

- The meaning of a sentence can be composed from the meaning of its parts, such as words and phrases.
- The meaning of a sentence is not based on the constituent words. It also depends on the ordering, grouping and relations among the words in the sentence.
- In syntax-driven semantic analysis, the composition of meaning is guided by the syntactic components and relations provided by grammars.

Overall Structure

- In the system, there is a parser to decide the syntactic structures, and a semantic analyzer to output the meanings.
- Ambiguity in lexicon and syntax will lead to multiple meanings. It is not the job of this narrowly defined semantic analyzer hereby to resolve such ambiguities.

An Example

- Consider the example

  AyCaramba serves meat.

- It has a parse tree from

  S $\Rightarrow$ *NP VP* $\Rightarrow$ *PropN VP* $\Rightarrow$ *PropN Verb NP*

- The target meaning representation is

  $\exists e$ *ISA(*e*,Serving)* $\wedge$ *Server(*e*,AyCamramba)* $\wedge$ *Served(*e*,Me*

- We want a method to go from the syntactic structure (parse tree) to the target meaning representation, for every possible input sentence.

Fundamental Questions

- What is the meaning of syntactic constituents?
- What do the meaning of smaller units look like so that they can be composed into meanings for a larger unit?
- Since sentences are derived from lexicon and syntax, the places to look at are the lexicon entries and grammatical rules.

Semantic Attachment

- Semantic attachment to a CFG rule specifies how to compute the meaning of a larger unit given the meaning of some smaller parts.

$$A \rightarrow \alpha_1 \ldots \alpha_n \ \{f(\alpha_j.sem, \ldots, \alpha_k.sem)\}$$

- The semantic attachment of the constituent A, denoted by $A.sem$, is governed by the function f.
- It is a function of the semantic attachments of the constituents.

Nouns

- The simplest case is to attach constants with the trees that introduce them,

  ProperNoun $\rightarrow$ *AyCaramba {AyCaramba}*

  MassNoun $\rightarrow$ *meat {meat}*

- The semantic attachments of NPs can be defined by

  NP $\rightarrow$ *ProperNoun {ProperNoun.sem}*

  NP $\rightarrow$ *MassNoun {MassNoun.sem}*

- In general, the semantic of child is copied to its parent if it is non-branching for NP.

Verb

- We have the semantics of constants (Nouns). For the verb *serves*, there is a *server* and something *served*,

$$\exists e, x, y \; \textit{ISA}(e,\textit{serving}) \wedge \textit{Server}(e, y) \wedge \textit{Served}(e, x)$$

- That logical formula can be the semantic attachment for

$$\textit{Verb} \rightarrow \textit{serves}$$

The Example

- What is the semantics for the subtree rooted *VP*?
- Here we have two constituents *NP*, *Verb*, and we want to express the semantics of *VP* as a function of *NP.sem* and *Verb.sem*.
- The target *VP.sem* is

$$\exists e, y \; \textit{ISA}(e,\textit{serving}) \wedge \textit{Server}(e, y) \wedge \textit{Served}(e,\textit{meat})$$

- We need a way to replace the variable x in *Verb.sem* by *NP.sem*. We use the so-called λ-notation.

λ-Notation

- The λ notation (or expression) is of the form

$$\lambda x P(x),$$

consisting of λ, one or more variables x, and an FOPC expression $P(x)$.

- In an FOPC formula, a variable x following λ can be replaced by a specific FOPC term, followed by the removal of λ and x. This is called λ-reduction.

$\lambda x P(x)(A) \Rightarrow P(A)$

$\lambda x \lambda y$ *Near*(x,y) *(ICSI)* $\Rightarrow \lambda y$ *Near(ICSI,*y*)*

λy *Near(ICSI,*y*) (AyCaramba)* $\Rightarrow$ *Near(ICSI,AyCaramba)*

Putting Together

- With λ-notation, the semantic attachment of Verb and VP can be written as

  Verb $\rightarrow$ *Serves*

  {$\lambda x \exists e, y$ *ISA(*e*,Serving)* $\wedge$ *Server(*e, y*)* $\wedge$ *Served(*e, x*)}*,

  VP $\rightarrow$ *Verb NP {Verb.sem(NP.sem)}*

- To complete the example, we need

  S $\rightarrow$ *NP VP {VP.sem(NP.sem)}*,

  which would have required us to modify the Verb attachment to include $\lambda x \lambda y$.

Variation

- Consider a similar sentence.

  A restaurant serves meat.

- If we follow the same procedure, we get the semantics

  $\exists e$ *ISA(*e*,Serving)*

  $\wedge$ *Server(*$e, \exists x$ *ISA(*x*, Rest))* $\wedge$ *Served(*e*,meat)*

- The Server-predicate is not a valid FOPC formula. In this case, we introduce the **complex term**.

Complex Term

- A complex term is an expression with structure

  < Quntifier Variable Body >

- To convert a complex term to FOPC, use

  $P($*< Quntifier Variable Body >*$) \Rightarrow$

  Quntifier Variable Body $Connective\ P($*Variable*$)$

- For example, the above case becomes

  Server(e*, <* $\exists x$ *ISA(*x*, Rest) >)*

  $\Rightarrow \exists x$ *ISA(*x*, Rest)* $\wedge$ *Server(*e, x*)*

Semantic Grammars

- A rule of semantic grammar looks like

$$\textit{InfoReq} \rightarrow \textit{User want to eat FoodType TimeExpr}$$

- It is combined with rules for FoodType, TimeExpr, User, and so on.
- The semantics is in the grammatical rule.
- Such a rule can be obtained from a corpus. It is very restricted indeed, and can be useful in very limited domain.
- As another example,

$$\textit{InfoReq} \rightarrow \textit{when does Flight arrive in City}$$

Information Extraction

- Information extraction tasks are characterized by two properties:
  - the desired knowledge can be described by a relatively simple and fixed template, with slots to be filled in with materials from text,
  - only a small part of the information in the text is relevant.
- For example, MUC-5 task requires systems to produce hierarchically linked templates describing participants, resulting company, intended activity, ownership and capitalization of a joint venture of business.

Lexical Semantics

- Meaning is associated with a sentence in our earlier discussion.
- Words, by themselves, cannot be judged to be true or false, literal or metaphorical.
- However, it is obvious that the meaning of a sentence is dependent on the *senses* of component words.
- Lexical semantics is the study of word senses.

Lexemes

- A **lexeme** has three components
  - orthographic form
  - phonological form
  - meaning component, called sense
- A **lexicon** is a list of lexemes.

Lexical Relations

- There are some interesting phenomenons about the definitions in a dictionary: they are descriptions rather than definitions.
- It is evident that there are relations between lexemes.
  - left, right
  - red, color
  - blood, liquid
- Word definitions are stated in terms of other lexemes. So circularity is unavoidable.
- A lot can be learned if we analyze and label the *relations* between lexemes.

Homonymy

- **Homonym** is a relation that holds between words with the same forms but unrelated meanings.

  financial *bank*^{1} vs. east *bank*^{2}

- Words of the same pronunciation but different spellings are not considered homonyms (be, bee). They are called *homophones*.
- Words of the same spelling but different pronunciations are not considered homonyms (CONtent, conTENT). They are called *homographs*.
- Lexemes with the same pronunciation and spelling but different POS's are not considered homonyms.

Applications Made Complicated

- spelling correction (homophones)
- speech recognition (homophones, homographs)
- text-to-speech (homographs)
- information retrieval (homographs)

Polysemy

- **Polysemy**: multiple related meanings with a single lexeme.
- The definition of lexeme is extended to include a set of *related* senses rather than a single sense.
- Consider blood *bank*, is this *bank* the same as *bank*^{1}? Obviously it is related.
- The difference between homonym and polysemy can be difficult to tell.

Polysemous Senses

- Given a single lexeme, we'd like to answer
  - What are the senses?
  - How are the senses related?
  - How can they be distinguished?
- Consider the word *serve* in

  They serve red meat.

  He serves as US ambassador to Japan.

  He served his time.

  How can one decide that they have different senses?

Synonymy

- Different lexemes with the same meaning are called **synonyms**.
- Whether two lexemes have the same meaning can be tested by **substitutability**.
- It is hardly possible for two lexemes to be interchangeable in all contexts. We often call two lexemes synonyms as long as they are interchangeable in some context.

Examples

- Whether one lexeme can be substituted for another depends on several factors. Here we give a few examples for substitution.

big/large plane (OK)

big/large sister (not OK)

first-class *fare/price* (odd)

big/large mistake (not OK)

Hyponym

- A kind of relation between lexemes is that one lexeme is a subclass of the other.
- The more specific lexeme is called a **hyponym**, and the more general lexeme is called a **hypernym**. For example

  car is a hyponym of *vehicle*

  vehicle is a hypernym of *car*
- The concept of hyponymy is related to notions in biology and computer science.

Ontology

- **Ontology** refers to a set of objects obtained from an analysis of a domain, called microworld.
- A **taxonomy** is an arrangement of the objects in an ontology into a *tree-like* class inclusion structure.
- **Object hierarchy** (of an ontology) is based on the notion that objects arranged in a taxonomy can inherit features from their ancestors.

Database of Lexical Relations

- It is clear that a database for lexical relations is very useful. **WordNet** is such a lexical database.
- In fact, it consists of 3 separate databases: nouns, verbs, adjectives and adverbs.
- A lexical entry in WordNet has a unique orthographic form (no phonological form), accompanied by a set of senses.
- Entries can be accessed directly with a browser or through a set of C library functions.

Note

- WordNet does not use phonological form in the definition of a lexeme.
- WordNet does not distinguish homonymy from polysemy. The distinction between polysemy and homonymy can be subjective and controversial.

WordNet Entry

- An entry in WordNet consists of an orthographic form and a set of senses. The lexeme *bass* is given in Figure 16.2.
  - There are 8 senses.
  - Some of the senses are clearly related (polysemy), while others are not (homonymy).
- The power of WordNet lies in its domain-independent lexical relations. The relations hold between lexemes, senses or sets of synonyms.

Noun Relations

Hypernym: *breakfast* $\longrightarrow$ *meal*

Hyponym: *meal* $\longrightarrow$ *lunch*

Has-Member: *faculty* $\longrightarrow$ *professor*

Member-Of: *copilot* $\longrightarrow$ *crew*

Has-Part: *table* $\longrightarrow$ *leg*

Part-Of: *course* $\longrightarrow$ *meal*

Antonym: *leader* $\longrightarrow$ *follower*

Relations of Verbs

Hypernym: *fly* $\rightarrow$ *travel*

Troponym: *walk* $\rightarrow$ *stroll*

Entails: *snore* $\rightarrow$ *sleep*

Antonym: *increase* $\rightarrow$ *decrease*

Synonyms

- Two lexemes are considered synonyms if they can be successfully substituted in some context.
- Synonymy is implemented in WordNet by **synset**.
- For example, the synset for *a person who is gullible and easy to take advantage of* is

  {*chump, fish, fool,* ...}

- A synset actually constitutes a sense, which is used in defining lexemes.

Synset

- Each synset represents a concept that has been lexicalized in the language
- To find chains of more general or more specific synsets, one can follow a chain of hypernym or hyponym relations.
- Different concepts follow different chains. Eventually they converge at **entity**, which basically serves as the top of conceptual hierarchy.

Thematic Roles

Agent: *The waiter spilled the soup.*

Experiencer: *John has a headache.*

Force: *The wind blows debris.*

Theme: *The waiter spilled the soup.*

Instrument: *The chef cut the fish with a knife.*

Beneficiary: *He booked a ticket for me.*

Source: *I come from Taiwan.*

Goal: *I am going to Japan.*

Selectional Restriction

- There are semantic constraints on filling the thematic roles of a verb by lexemes.
- Selectional restriction helps to disambiguate

  I want to eat someplace that's close to ICSI.

- The hyponym relation in WordNet can help to enforce selectional restrictions. A lexeme represents something edible as long as something up the hyponym chain is edible.

Metaphor

- With metaphor, we express some concept using words or phrases with completely different concepts.
- Metaphor is pervasive.
- Many metaphors are motivated by a small number of conventional metaphors. For example, the metaphor of organization-as-person, as in

  Microsoft says open-source violates 235 patents.

Metanymy

- Sometimes a concept is referred to by some closely related concepts. This is called metanymy.
- Musician-for-his(her)-works is a common type of metanymy

  He likes Mozart.

  Place-for-institution

  White House has no comments.
- Metaphor and metanymy poses challenges for lexical semantics.

Word Sense Disambiguation

- A word may have multiple senses. In a sentence, only one sense is accurate.
- Without context, it is impossible (absurd) to decide sense.
- The process of determining which sense is used given the context of a word is called word sense disambiguation.
- Conventional word sense disambiguation does not distinguish between homonymy and polysemy sensibly.

Information Retrieval

- The most successful application in this era. Billions of Google search everyday.
- Current IR systems are based on individual words without considering their grouping and ordering. Such a methodology is also called *bag-of-words*.

In a Nutshell

- A **document** is a unit for retrieval, which is indexed by an IR system.
- A **term** is a lexical item or phrases.
- A **query** is a set of terms. It is used by a user to retrieve documents.
- In an ad hoc information retrieval, a user inputs a query and the system returns a set of potentially useful documents to the user based on his query.

Vector Space Model

- Documents and queries are represented as vectors in a space.
- A component of the vector indicates whether a specific term is present or not in the document or query.

$$d_{i,j} = \begin{cases} 1, & \text{if } t_i \text{ is in document } j \\ 0, & \text{if } t_i \text{ is not in document } j \end{cases}$$

- A *term-by-document matrix* is defined by d_{ij}, where each column represents a document.

Term Weighting

- Terms limited to a few documents should be given more weights, while terms common to most documents should be given less weights. Such consideration leads to *inverse document frequency* term weight

  $$\text{idf}_i = \log \frac{N}{n_i},$$

  where N is the number of documents, n_i is the number of documents containing term w_i.
- *Term frequency* tf_{ij} is the number of occurrences of term w_i in document d_j. A common term weighting is

  $$w_{ij} = \text{tf}_{ij} * \text{idf}_i$$

Weighted Vector

- Without term weighting, a document (or query) is represented by

  $$d_j = (t_{1j}, t_{2j}, \ldots, t_{Mj}),$$

  where M is the number of lexical items in the text collection.
- With term weighting, such as the scheme defined in the previous slide, the document vector becomes

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

Similarity Measure

- The similarity of two vectors can be measured by their inner-product, or the angle between them.

$$sim(q_k, d_j) = \sum_i w_{ik} w_{ij}$$

$$sim(q_k, d_j) = \frac{\sum_i w_{ik} w_{ij}}{|q_k||d_j|}.$$

Precision and Recall

- **Recall** measures the system's ability to retrieve relevant documents from the collection

$$R = \frac{\text{\# of relevant documents returned}}{\text{total \# of relevant documents in the collection}}$$

- **Precision** measures how likely are the documents returned by an IR system are actually relevant.

$$P = \frac{\text{\# of relevant documents returned}}{\text{total \# of returned documents}}$$

F-measure

- A loose rule leads to high recall and low precision, while a strict rule leads to low recall and high precision.
- To balance, F-measure is often used. It is related to recall and precision with a parameter β

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

- Setting $\beta = 1$ treats P and R symmetrically.

Stemming and Stop List

- With stemming, *processing, processed, processes* are treated as the same term.
- But *stock, stockings* are also treated as the same term.
- Stemming increases recall and reduces precision.
- A stop list is a list of high frequency words that are ignored in the vector space model.
- Stop list ignores function words and thus saves space and time.

Word Senses

- Homonymy and polysemy have the effect of reducing precision since usually only one of the senses is relevant yet the system may return documents related to other senses.
- Synonymy and hyponymy, on the other hand, may have the effect of reducing recall as the system may miss related documents containing synonyms or hypernyms of the query terms.
- Does word-sense disambiguation help IR? Mixed results.

Relevance Feedback

- A user specifies whether returned documents are relevant to his need.
- Distribution of terms in relevant and irrelevant documents are used to reformulate query.
- Intuitively, we want to *push* the query towards (away from) the relevant (irrelevant) documents in the vector space. This can be achieved by adding (subtracting) an average vector to the query.

$$q_{i+1} = q_i + \frac{\beta}{R} \sum_{i=1}^{R} r_i - \frac{\gamma}{S} \sum_{k=1}^{S} s_k$$

Query Expansion

- The query is expanded to include terms related to the original terms.
- Highly correlated terms can be found in a *thesaurus*.
- A suitable thesaurus can be generated automatically from a collection of documents.
- One common method for thesaurus generation is *term clustering*. The rows of the term-by-document matrix can be clustered.

Other IR-related Tasks

- Document categorization: assign a new document to a predefined set of classes.
- Document clustering: discover a reasonable set of clusters for a given set of documents.
- Text segmentation: break larger documents into smaller coherent chunks.
- Text summarization: produce a shorter, summary version of an original document.