NLP

1. Introduction

NLP paradigms

- Symbolic approaches
  - knowledge intensive, rule based and hand coded
- Statistical approaches
  - Distributional and neural approaches. Supervised or unsupervised
  - Data intensive

There are several different fields of NLP applications. Those include one with generating natural language text, or analyze the natural language, understand the natural language and response.

2. Basic Text Processing

Intro

- Polysemy : one word maps to many concept
- Synonymy : one concept maps to many words
- Word order has a significant impact on concept of words.
- Language is generative : short sentence and long sentence can mean same thing in general
- Language changes over time.

- ill formed text input : text can contain any sort of error such as typo
- Multi linguality : the same word can refer to totally different things in different language
- Sarcasm, irony, slang, jargo

Fundamentals

- Language is ambiguous
- Lexical analysis (tokenization)
- Stop word removal
- Stemming : chop the end off. Not work with every words.
- Lemmatization : linguistically principled analysis. Not work with every words.
- Morphology ?
- Syntax : tags on words.
  - Noun, Pronoun, Adjective, Determiner, Verb, Adverb, Preposition, Conjunction, Interjection
- Ambiguity problem
- Parsing (grammar)
  - often use tree like structure
- Sentence boundary detection
  - punctuation is not necessary the case, there are other way to end sentence

Basic Text Processing

- Sentence Segmentation : where a sentence begin / end
  - nltk provide useful functions / tools
- Word tokenization
  - in most cases, words can be tokenize by a space

- there are some cases that space isn't the best way to tokenize, such as punctuation, compounds, contractions, irregular words including dates, email, web addresses

- Text normalization
  - To relate variation of words to a common form / root
  - Both stemming and lemmatization
  - Need to consider both inflection forms and derivational forms
    - Inflection forms (morphology) : different forms of the same grammatical category
      - eg. run, runs, running ⇒ run
    - Derivational forms (morphology) : different forms across different grammatical forms
      - eg. democratic and democracy
- Regular expression
  - a formal language for defining text strings (character sequence)
  - used for pattern matching
    - eg. searching and replacing in text

- Stop words
  - high frequency words with little lexical meanings
    - eg. and, the, of, a
  - it can possibly add some noise for ML trining, but not always
  - filter out beforehand
  - language specific, no universal list of stop words

- Text corpora
  - essentially a large list of text / large collection of text data

    - there are several text corpora in nltk

3. Language Modeling

Data analysis on text data

- word type and word token
    - word tokens refer to every words on text
        - a length of text
    - word type refer to number of vocabulary
        - a length of text without duplicates
- lexical diversity : len of word type / len of word token
- Frequency distribution
    - how many times each word types occur in text
- collocations
    - words that are ‘stick together’
- conditional frequency distribution
    - frequency distribution on certain category such as title, author, topic genre
- bigrams
    - a pair of words in a sequence

Probabilistic Language Modeling

- This language model uses statistical probability to model language. For instance, there are different sequence of words that can possibly have the same meaning, but one of them is more likely to be correct (in machine translation). Another instance where two sentences have a same or very similar phonetics, correct sentence can be generate by picking words that are statically likely to be correct.

- Computing probabilistic
  - Probability of sequence : $P(w) = P(w_1, w_2, w_3, ..., w_n)$
  - Probability of upcoming term : $P(w_1|w_2, w_3, w_4)$
  - Chain rule in probability theory : $P(x_1, x_2, x_3, ..., x_n) = P(x_1)P(x_2|x_1)P(x_3|x_2, x_1)...P(x_n|x_1...x_{n-1})$

  Probabilities of above are essentially looking at the whole text data, and is limited on that data set. In this case, the model will say that some of sentence as 0% of probability because it never appear on the data set.

  So, it is better to look at much smaller segment of text data.

  - Bigrams Frequency : $P(w_i|w_1, w_2, ..., w_{i-1}) = \prod_i (w_i|w_{i-1})$
  - Unigram Frequency : $P(w_1, w_2, ..w_n) = \prod_i P(w_i)$

    *note

    The capital Greek letter Π (capital Pi) is used in math to represent the product operator. Typically, the product operator is used in an expression like this:

    $$\prod_{i=0}^{2} [i+1] = 1 \cdot 2 \cdot 3$$

    In plain language, this means take the product of the sequence of expressions represented by the expression $i+1$ starting from $i=0$ and iterating until $i=2$.

  - Estimate second order (bigram) probabilities using a maximum likelihood estimator.

    $P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$ ($c()$ refers to a count of words)

  - There are still a case where possible sequence being 0 because it doesn't appear on the data set.

    Let's say there are two sentences ...

    - the cat sat on the mat

- a dog sat on the cat

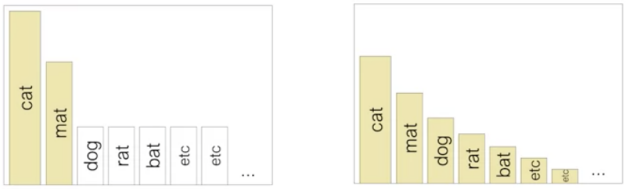
then...

dog never appear after ‘the’ but it is possible...

The sparse data problem

We need to ‘**smooth**’ the probability distribution

P(W|the)
2 * cat
1 * mat
0 * dog
etc.



- Laplace smoothing : $P(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+1}{c(w_{i-1})+V}$ (V : vocabulary)
- In fact, often this is too crude approach and more advanced techniques are used.

Backoff & interpolation

Backoff: use **smaller contexts** where there is **less evidence** (training data)
e.g. trigram -> bigram -> unigram

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases} \qquad S(w_i) = \frac{\text{count}(w_i)}{N}$$

Interpolation: mix unigram+bigram+trigram:

$$\hat{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n) \qquad \sum_i \lambda_i = 1$$

Evaluate Language Model

- Language Model should prefer ‘likely’ sequence over ‘unlikely’ ones
  - like any other machine learning model, it should train the model with training data set and test / validate the model with unseen data set.
- Two approaches
  - Extrinsic : use / test in real app and measure performance
    - it is expensive
  - Intrinsic : use perplexity
    - Perplexity
      - analogous to the branching factor
      - the best LM is the one that best predict the test set
      - perplexity is the inverse probability of the test set
      - high probability ~= low perplexity

$$PP(w) = P(w_1, w_2, ..., w_n)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1, w_2, ..., w_n)}} = \sqrt[N]{\Pi_{i=1}^{N} \frac{1}{P(w_i | w_1, ..., w_{i-1})}}$$

Topic Modeling

video from 3.3

4 : Lexical Semantics

Introduction

- The meaning of words

- NLP task
    - Classification tasks
    - Sequence tasks
    - Meaning tasks
- Useful for
    - Information retrieval
    - Question answering
    - Topic modeling
- Dictionary is one way to store the meaning of words
- Another way to organize the meaning of words ....
    - Lexical database (WordNet)
        - Nodes are synsets (a set of synonyms)
        - Correspond to abstract concepts
        - Polyhierarchical structure
    - Using WordNet, we can programmatically identify hyponyms (child terms) and hypernyms (parent terms). Also, measure semantics similarity

Intro to Word Embedding

WordNet requires human effort to maintain and build

Words with similar context have similar meanings

- “If A and B have almost identical environments we say that they are synonyms” - Zelling Harris (1954)
- The distribution hypothesis
    - Distributional models are based on a co-occurrence matrix
        - term-document matrix, term-term matrix

        - Term - document matrix
            - overall matrix : |V| by |D| (V : vocabulary D: document)
            - Similar documents have similar words (column)
            - Similar words occur in similar documents (row)
        - Term - term matrix
            - overal matrix : |V| by |V|
            - represent co-occurrences in some corpus
            - context is usually restricted to a fixed-window
            - similar terms have similar vectors

    - Word Embedding : represent words using low-dimentional vector
        - Capture the similarity between terms
        - 50 - 300 dimensions
        - most values are non-zero (dense)
        - benefits are...
            - classifiers need to learn far fewer weights
            - helps with generalization, which avoid overfitting
            - capture synonymy
            - easier to use with ML
    - Other statics embedding
        - fasttext : deal with unknown words and sparsity by using subword models
        - GloVe : uses global corpus statistics. combine count-based models with word2vec linear structures

5. Text Categorization and Sentiment Analysis

Examples of text categorization

- Authorship attributes, subject classification, language identification, sentiment analysis and etc (video 5.1)

- text categorization methods
  - Inputs are usually,,,,
    - A document d
    - A fixed set of classes $C = \{c_1, c_2, ..., c_n\}$
    - A set of example documents $(d_1, c_1), (d_2, c_1), (d_3, c_3), (d_4, c_2).....(d_m, c_n)$
  - Output will be
    - A classifier $\gamma : d => c$
  - Classifier can be rule based. It can have high accuracy but difficult to scale and maintain. Requires lots of human power and resources
  - ML classification is more efficient
    - It aim to predict discrete label of a document
    - Find a line that separate the classes. (Learn the parameter of the line)
    - There are variety of techniques including Naive Bayes, Logistic regression, Support vector machines, neural approaches

Bayes Theorem

Naive Bayes

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)}$$

$P(H)$: Prior $P(E|H)$: Likelihood $P(H|E)$: Posterior

$P(E)$: Model evidence or Marginal likelihood

video from 5.1

- Evidence should be used to update your beliefs, not determine them
- Can be used whenever
  - Hypothesis (H)
  - Evidence (E)
  - Want to determine $P(H|E)$
- Conceptualise in terms of areas rather than counts
- For classification, we can think of Bayes Theorem in terms of features and labels

$$P(L|Features) = \frac{P(L)P(Features|L)}{P(Features)}$$

- To chose between labels, we can compare their posterior probabilities

$$\frac{P(L_1|Features)}{P(L_2|Features)} = \frac{P(L_1)P(Features|L_1)}{P(L_2)P(Features|L_2}$$

- Bayes Theorem for classification
  - Need to calculate $P(Features|L_i)$ for each label
  - Assume a generative model for each label ⇒ features
    - Specifying this model is the main part of training in Bayesian classification
  - Apply some simplifying (’naive’) assumptions
    - Independence
    - Equality
  - Naive Bayes Classification
    - First and simple classification algorithm
    - high dimensional database
    - Few tunable parameters : often use as base line
- Multidimensional Naive Bayes

- Features are generated from a multidimensional distribution
  - Probability of observing counts across various categories
  - Find a best-fit multinominal distribution
- Examples ....
  - Text classification
  - Categories are word types
- Convert from strings to numbers ...

Semantic Analysis

- Common application of semantic analysis
  - reviews on various products
  - prediction
- Scherer typology of different state
  - Personality traits : stable personality dispositions and typical behavior tendencies
    - Nervous, anxious, reckless, morose, jealous etc
  - Mood : diffuse non-caused low-intensity log-duration change in subjective feeling
    - cheerful, gloomy, irritable, listless, depressed,etc
  - Interpersonal stances : affective stance toward another person in a specific interaction
    - friendly, flirtatious, distant, cold, warm, supportive, etc
  - Attitude : enduring, affectively colored beliefs, dispositions towards objects or person
    - liking, loving, hating, valuing, desiring
- Basic sentiment classification
  - It is the detection of attitudes

- simple task : is the attitude of a giving task positive or negative
- Naive Bayes approach:
    1. Estimate prior
    2. Ignore unknown words
    3. Estimate likelihoods
    4. Compare score for each class

- Optimizing for sentiment analysis
    - Occurrence matters more than frequency
        - Use binary multinominal Naive Bayes
        - limit word count to one
            - eg. the idea is good so is the execution → the idea is good so execution
    - Negation changes polarity
        - eg. The book was good → The book was not good
            - word ‘not’ change the polarity of the sentence
        - eg. Overall it’s not bad.
            - Sentiment is different from previous sentence
        - Simple baseline method : add NOT_ to every word between negation and following punctuation.
            - eg. I didn’t like this book, but → I didn’t NOT_like NOT_this NOT_book, but
    - Sentiment lexicon
        - Manually created wordlist, it is useful when training data is limited
        - video on 5.2

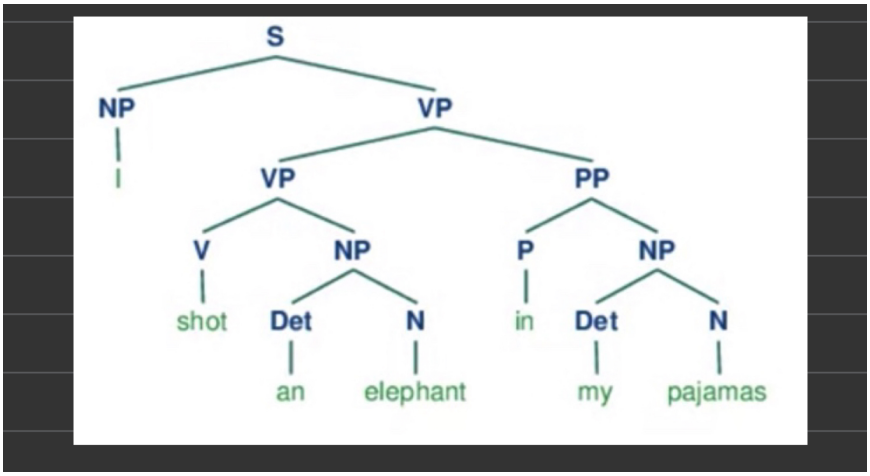
6. Syntax and Parsing

Syntax : The way in which words are arranged.

Constituency : Groups of words behave as one units. Grammar refers to inventory of constituents for a language

Context-free grammars (phrase-structure grammars)

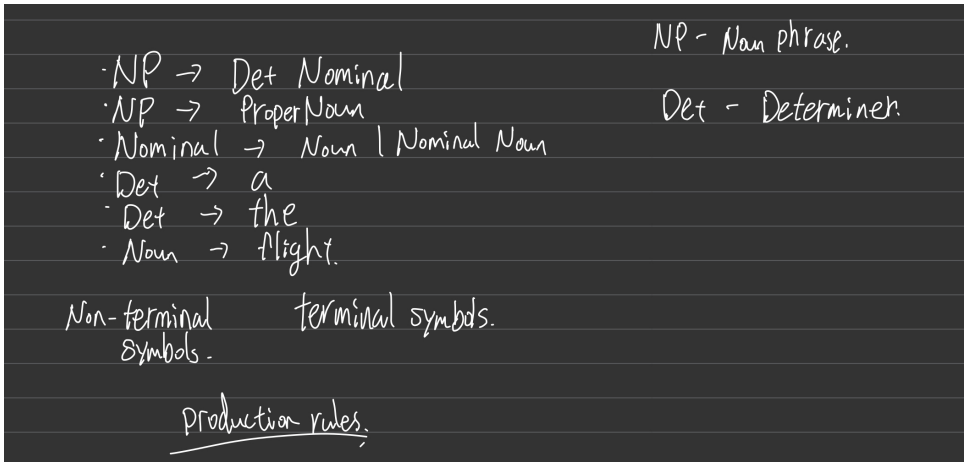
- A set of rules
- A lexicon of words or symbols
- Can be used for generation or assigning structure . Structure is commonly represent as a parse tree.

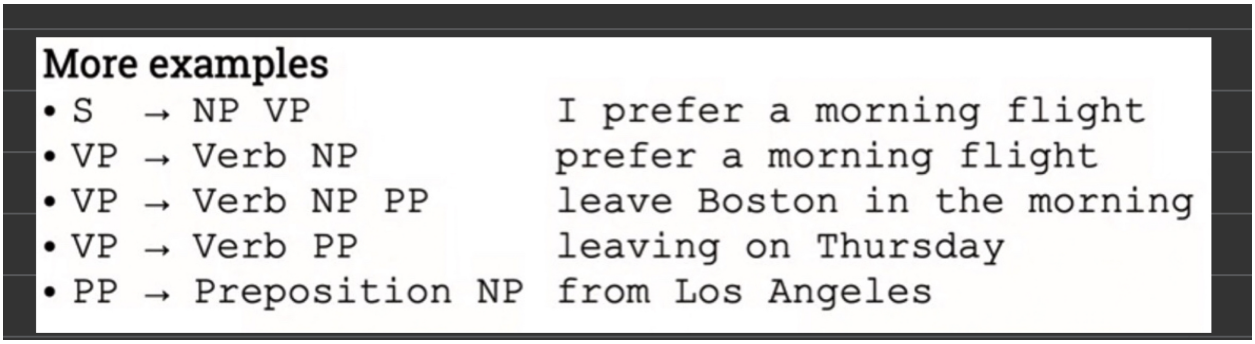


Every grammar has a start symbol which is usually the S node which refers to sentence

- Language : set of strings that are derivable from start symbol
- Sentence level construction (for English)
  - Declarative / imperative / yes-no / wh-structure

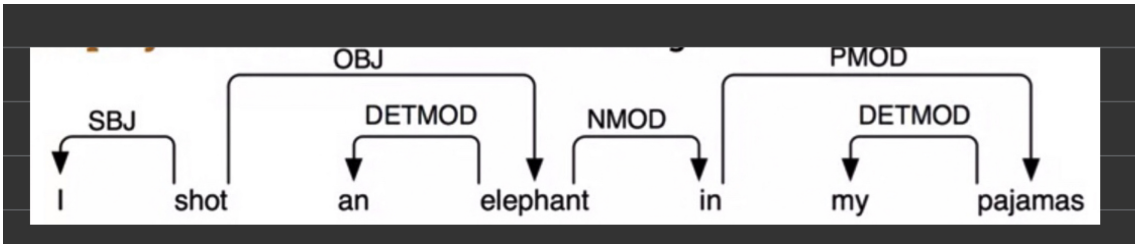
Examples :





Dependency grammar

- Phrase structure focuses on how words combine to form constituents. Dependency grammar focuses on how words relate to each other.
- Dependencies are binary asymmetric relationships
  - Between a head and its dependents : labelled directed graph
  - Head is usually a tensed verb
  - Arcs are dependency relations (head → dependents)
  - Graph is projective if there are no crossing



7. Information Extraction

Information extraction is a process of extracting semantic content from text which is often a factual information. It is a process of taking unstructured information and return structured information. Also, it is process of converting strings into database entities.

- Unstructured information → Structured information
- Strings → Database entities

- Subtasks within information extraction
  - Named entity recognition : what is being talked about
  - Relation extraction : how they are connected
  - Event extraction : events in which the entities participate
  - Event coreference : resolving ambiguous mentions
  - Temporal expression extraction : when the above event happened

Relation extraction algorithms

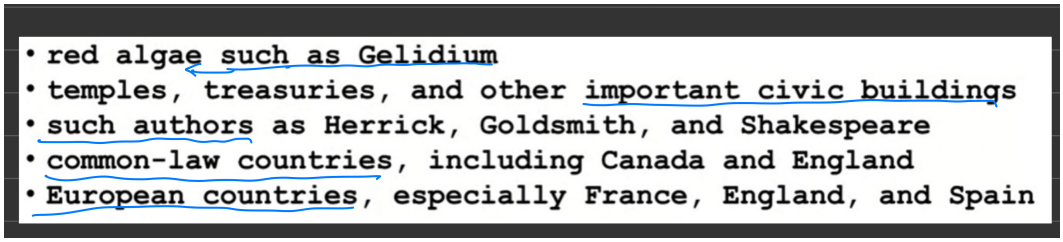
1. Handwritten patterns (few decades old)
   - Infer a hyponym relation

     NP_0 such as $NP_1\{NP_2....(and \mid or)NP_n\}\ ,\ \ i >= 1$

     Implies this semantics:

     $\forall NP_i,\ i >= 1 \quad hyponym(NP_i, NP_0)$

examples



can add other constrains such as entity type

- Pros : High precision, domain specific
- Cons : low recall, not scalable

2. Supervised learning
   - Choose entities and relation → Annotate a training corpus → Train classifier
   - Runtime : find a pairs of NEs, then apply classifier to each pair. Can pre-filter to identify pairs with any relation (NE : Named Entities)
   - Sample features for each entity instance
     - Word features : headwords, bag-of-words, bigrams
     - NE features : entity types, entity level
     - Syntactic structure : constituent paths, dependency path
   - It can also use neural approaches
     - Use pre-trained encoder (eg BERT)
     - Add linear on top, fine-tuned as 1-of-N classifier
     - Runtime : Apply classifier to pair of entities in inputs, can release entities by NER tags to avoid overfitting
   - Pros : High accuracy, domain-specific
   - Cons : Expensive, brittle
3. Semi-supervised learning

   1) Bootstrapping

- Strat from high precision seed pattern or tuple
- Find sentence that contain them
- Generalize the context around them to generate new patterns

2) Distance supervision

- Start from large database of seed patterns or tuples
- Creates lots of noisy patterns
- Combine them into a supervised classifier

...

4. Unsupervised learning

- No labeled training data , open information extraction
- Relations are just strings of words (usually beginning with a verb)
  - Run a POS tagger and entity chunker over each sentence
  - Find longest phrase that starts with a verb
  - Find nearest NPs to the left and right
  - Assign confidence c to relation
- Relations are stored if they occur with a least N argument

8. Information Retrieval (IR)

IR

Find a documents that satisfy an information need

- Document can be anything such as emails, personal records, products etc
- Search within unstructured information
- Searching in large collections which makes it challenging and interesting problem
- Documents are not database data, it is strings and such that are unstructured

Use case

- Professional search : document databases in legal, healthcare etc
- Enterprise search : various repositories within organization
- Site search : catalogs in eCommerce, media, SNS etc
- Web search : Yahoo, Google, Bing etc

Information needs are typically task related, but not always

Models of IR process

- Classic model
- Standard model
- Dynamic model

Using term-document matrices

- With huge amount of data, it will be a very big size of matrices

The inverted index

- Store a list of docIDs for each term t
- Variable sized posting lists, stored by docID
- Set of terms is the vocabulary
- Typical pipeline
    - Tokenization → Normalization (conflate various forms) → Stemming / lemmatization → stop word removal
- Query processing : Boolean operations

Boolean retrieval offers :

- Great control which means precise queries
- Transparency : you can see why documents were retrieved
- Reproducibility : same search should produce the same result. important characteristic for scientific research
- But it is not good for casual users :

- Don’t want to learn boolean
- Want to have a result in order

Ranked retrieval

- Return matching documents with a sorted order
- Query terms are just keywords
- Just sho the top N results

Bag of words model

- Taking all content from document or query and word order is ignore
- Weight each terms by term frequency
  - Terms that appears several times are probably important. But a term appears 10 times are not necessary 10 times more important than other terms
  - Usually take log of frequency

Inverse document frequency

- Rare terms are more informative than frequent terms
  - Greater discriminatory power → higher weight
- Document frequency df_t is the number of documents that contain term t.

TF.IDF

Product of TF (term frequency) and IDF (inverse document frequency)

- Frequency
- Informativeness

Space vector model

- Consider term document matrix, each document is a count vector (columns of length v), and terms are dimension which gives |V| dimensional space.
- Queries can also be represented in the same |V| dimensional space
  - It can rank documents according to their proximity to the query
- How to measure proximity?
  - General way to measure distance between two points is too large such as Euclidean distance
  - Rank documents by angle between vectors
    - Inverse of the angle or cosine
- Length normalization
  - Divide each component by its length using the LS norm $\sqrt{\Sigma_i x_i^2}$
    - makes it a unit vector on surface of unit hypersphere
    - d' and d are now identical

$cosine(query, document)$ is now

$$\frac{\Sigma_{i=1}^{|V|} q_i d_i}{\sqrt{\Sigma_{i=1}^{|V|} q_i^2} \sqrt{\Sigma_{i=1}^{|V|} d_i^2}}$$

q_i is TF.IDF weight for term i in query

d_i is TF.IDF weight for term i in document

For length normalization vectors, cosine is dot-product $\Sigma_{i=1}^{|V|} q_i d_i$

Summary

- Represent the query as a weighted TF.IDF vector
- Represent each documents as a weighted TF.IDF vector
- Compute $cosine$ score of each pair

- Rank documents by score
- Return top N for user

9. Dialogue Systems

Properties of human conversation

It is important to understand the human conversation in order to create dialogue system

There are some rules :

- A dialogue is a sequence of turn. One talk to other at a time
- Need to know when to stop / start talking. (detect an interruption and completion of turn)
- There are different actions performed by speaker : speech act
  - Constatives : commit to some proposition (claim, confirm, deny etc)
  - Directives : get the addressee to do something (ask, invite, etc)
  - Commissives : commit to some future course of action (promise, plan, vow, etc)
  - Acknowledgement : express reaction to some action (greet, thank, apologize, etc)

Grounding

- Dialogue isn’t just a sequence of turns. It is a collective act towards a common goal, characterized by cooperation
- Grounding : establish common ground (goal) between the participants
  - Acknowledging that the hearer has understood the speaker
  - Analogous principle in human-computer interaction : system feedback
    - Explicit : ‘ok’, ‘right’ , etc

        - Implicit : repeat back, some portion or important input of what speaker said
    - Dialogue can have nested structure : sub-dialogues
        - Correction : correct miss understanding occurs in conversation
        - Clarification

Initiative

- Dialogue may be controlled by one participant
    - eg. during interview, interviewer initiate the conversation
    - Usually, dialogues are mixed-initiative
    - Put single initiative is much simpler to offer
- User-initiative dialogues
- System-initiative dialogues
- Dialogue rely on inferences and conversational implicature
    - Heuristics that guide the interpretation
    - Maxims eg relevance

Chatbots

A simple dialogue system

- As opposed to task-oriented dialogue system
- it is more for entertainment purpose

Two classes of architectures

- Rule-based (ELIZA, PARRY)
- Corpus-based : mine large datasets, use IR techniques or encoder-decoder system

ELIZA : very first chatbots published on 1966 by Wiezenbaum, rule-based (pattern matching)

Corpus-based chatbots

- Data-driven approaches
  - Train on large corpus of huma-human conversation which can be sourced from call-centre dialogues, movie dialogues etc
  - It can be pre-trained on large datasets or exploit live-usage data
  - Confidence matrices needed
  - Responses generated through retrieval or generation method
  - Response by retrievals :
    - user's turn ⇒ query q retrieve highest matching response r from corpus C
    - Classic IR : TF.IDF + $cosine$

      $response(q, C) = argmax_{r \in C} \frac{q\ r}{|q|\ |r|}$
    - Neural IR : bi-encoder (one for q and one for r), take dot p[roduct of vector
    - Can consider all of previous conversation
  - Response by generation
    - Encoder-decoder task, transduce from user turn to system turn. Analogues to translation
    - Generate each token r_t by conditioning on entire query and response so far

      $\hat{r}_t = argmax_{w \in V}\ P(w|q, r_1 ..., r_{t-1})$
    - Form query from entire conversation so far
    - Downside is that it is somewhat predictable → Introduce elements of diversity
    - Response by retrieving and refining knowledge

        - Exploit external knowledge sources (eg. Wikipedia) + query expansion technique
        - Hybrid architectures also possible
            - template + regexes + classifiers

Task-based dialogue system

Unlike chatbot, its purpose is to help user to achieve a certain goal

- Frames : structured representation for user input
- Slots : attribute-value pairs in the frames

Process : ask questions, fill slots, perform relevant actions

- It can fill out multiple slot with one utterance
- Some task require multiple slots

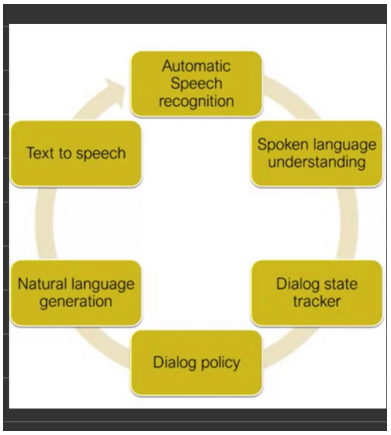
Three tasks of natural language understanding

- Domain classification
- Intent determination
- Slot filling

Matchings are done using rules (semantic grammars, regexes etc)

- High precision
- Expensive, difficult to scale, can be low in recall

Dialogue state architecture



- More sophisticated approach with dialogue acts and ML
- Dialogue state tracker : most recent act, all slots + fillers
- Dialogue policy : what system should do / say next.
  - when to answer user's questions v.s. when to clarify, suggest, etc
- Natural language generation : templates v.s. precise context
- Dialogue acts : specify the function of turn or utterance
  - Speech acts + grounding
  - task-specific tag set
- Slot filling
  - Supervised semantic parsing
  - Map from input words to slot fillers, domain and intent
- In practice, often bootstrap using GUS-style rules
  - Starts from hand-annotated rules & test set
  - New utterances are labelled using rules to produce new tuples
  - Classifier trained on tuples, evaluated on test set
- Dialogue state tracker needs to determine ....
  - Current state of frame
  - User's most recent dialogue act

        - Interpretation vis supervised classification based on hand labelled output
    - Dialogue policy
        - Decide what action to take next, based on entire dialogue state :

          $$\hat{A_t} = argmax_{A_i \in A}\ P(A_i | A_1, U_1, ..., A_{i-1}, U_{i-1})$$
        - Simplify by considering only current Frame and last exchange :

          $$\hat{A_t} = argmax_{A_i \in A}\ P(A_i | Frame_{i-1}, A_{i-1}, U_{i-1})$$
    - Natural language generation
        - Content planning : what to say
            - It is done by dialogue policy
            - Select dialogue out + some attributes slot + fillers
        - Sentence realization : how to say it
            - Trained on representation / sentence pairs from labeled corpus.
            - Use delexicalisation to generalize
            - Use encoder-decoder to map from frames to delexicalized sentences
            - Use input frame from content planner to relexicalize

Design Dialogue Systems

Unlike other NLP tasks, it is highly interactive, so adopt principles from human-computer interaction

User-centered design process :

1. Study user and tasks
2. Build simulations and prototypes
3. Interactively test and design on users

Dialogue system is very expensive to build even a prototype

Solution is to use Wizard-Oz methods to simulate interactions

- Can use output as a training data
- Method to emulate the dialogue system without actually building one by putting human researcher

Evaluating chatbot

- To evaluate chatbots, automated methods are not a good proxy. Manual approaches are better.
- Participant evaluation : 6 turns, 8 dimensions
    - Avoiding repetition, interestingness, making sence, fluency, listening, inquisitiveness, humanness, engagingness ....
- Observer evaluation : 3rd party annotators evaluate complete conversation, compare with benchmark system
    - engagingness, interestingness, humanness, knowledgeable
- Other NLP tasks often use automated testing, but chatbots are often in more diverse situations that doesn’t apply

Evaluate dialogue system

- Simple approach : measure task success
    - or alternatively...
        - User satisfaction (via questionnaire)
        - Slot error rate (or slot precision / recall / F)
- Ethical issues
    - ML systems replicate bias in training data
    - Gender stereotypes
    - Microsoft’s Tay chatbots (?)
    - Home dialogue agents & privacy implications