Foundations of Artificial Intelligence

Unit 5

Communication, Perceiving and Acting

Natural Language Processing:- Language Model - Text Classification - Information Retrieval - Introduction - Robot Hardware - Robot Perception - Planning to move - Application Domains

* Reasons to do Natural Language Processing

(i) For computers to communicate with humans

(ii) For computers to learn from human knowledge

(iii) To advance scientific understanding of languages and language use.

* Language Model

→ The use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occuring in a sentence.

→ LMs analyze bodies of text and provide a basis for their word predictions

→ Large language models also use language modelling

→ LMs are used in NLP, natural language understanding (NLU) and natural language generation (NLG) systems.

* Working of Language Models

→ LMs determine word probability by analyzing text data

→ They interpret this data by feeding it through an algorithm that establishes rules for context in natural language.

→ Then the model applies those rules in language tasks to accurately predict or produce new sentences.

→ The model learns the features & characteristics of basic language and uses those features to understand new phrases.

* Types of Language Models

A. N-gram models

→ This approach creates a probability distribution for a sequence of n

→ n can be any number, and defines the size of the gram, or sequence of words, or random words being assigned a probability.

→ This allows the model to accurately predict the next word or variable in a sentence.

→ If n=5, a gram may be something like: 'can you please call me?'

→ The model then assigns probabilities using sequences of n size.

→ n is essentially the amount of context the model is told to consider.

Mathematical representation for n-gram character models

→ $P(c_{1:N})$ = probability of a sequence of N characters c_1 to c_N

→ If the gram size is : one = unigram

two = bigram

three = trigram

→ A model of the probability distribution of n-letter sequences is thus called an n-gram model.

→ An n gram model is defined as a Markov chain of order n-1

→ In a Markov chain, the probability of character c_i depends only on the immediately preceding characters. So for a trigram model (Markov chain of order 2), it would be:

$$P(c_i | c_{1:i-1}) = P(c_i | c_{i-2:i-1})$$

→ The probability of a sequence of characters $P(c_{1:N})$ under the trigram model would be:

$$P(c_{1:n}) = \prod_{i=1}^{N} P(c_i | c_{1:i-1}) = \prod_{i=1}^{N} P(c_i | c_{i-2:i-1})$$

Language Identification with n-gram models

→ A trigram character model of each candidate language is built

$$P(c_i | c_{i-2:i-1}, l),$$

where the variable ranges over languages.

→ For each l, the model is built by counting trigrams in a corpus of that language.

→ Apply Bayes's rule followed by the Markov assumption to get the most probable language.

$$l^* = \underset{l}{\operatorname{argmax}} P(l \mid c_{1:N})$$

$$= \underset{l}{\operatorname{argmax}} P(l)\, P(c_{1:N} \mid l)$$

$$= \underset{l}{\operatorname{argmax}} P(l) \prod_{i=1}^{N} P(c_i \mid c_{i-2:i-1}, l)$$

Disadvantages of n-gram models

→ The training corpus only provides an estimate of the true probability distribution.

→ For eg. 'th' would give a good estimate, but 'ht' would not, even though there are words like HTTP.

Smoothing

→ The process of adjusting the probability of low-frequency counts is called smoothing.

→ The most simplest approach proposed by Laplace - an estimate for less frequent words should be $P(X = true) = \frac{1}{N+2}$

Other approaches are:

→ Backoff Model - start by estimating n-gram counts, but for a sequence that has low or zero count, back off to n-1 grams

→ Linear Interpolation smoothing - a backoff model that combines trigram, bigram & unigram models by linear interpolation:

$$P(c_i \mid c_{i-2:i-1}) = \lambda_3 P(c_i \mid c_{i-2:i-1}) + \lambda_2 P(c_i \mid c_{i-1}) + \lambda_1 P(c_i) \quad \Big| \quad \lambda_1 + \lambda_2 + \lambda_3 = 1$$

B. Bidirectional

→ can analyze text in both directions, both forward and backward

→ can predict any word in a sentence or body of text by using every other word in the text.

→ Examining text bidirectionally increases result accuracy.

→ used in ML models and speech generation applications

C. Exponential

→ It models the probability of a word appearing in a given context.

→ It is a statistical model based on the idea that the probability of an event is proportional to the probabilities of its individual features

→ It evaluates text using an equation that combines feature functions and n-grams.

→ The more features an event has, the more likely it is to occur.

D. Continuous Space

→ represents words and their relationships in a continuous vector space.

→ useful especially as the dataset gets bigger, as larger datasets often include more unique words

→ Words with similar meanings are represented by vectors that are close together in the space.

→ allows the LM to capture the nuances of language & generate more natural and fluent text.

* Natural Language Toolkit - NLTK

→ A python library for preprocessing human language data

Features

(i) Ease of use - a simple and intuitive interface for performing NLP tasks such as tokenization, stemming and POS tagging.

(ii) Large collection of data and resources - NLTK includes a large corpora, and resources for working with them, such as lexicon, grammars, and annotated corpora.

(iii) Support for various languages - supports many languages and tools for working with them.

(iv) Compatibility with other libraries - compatible with other Python libraries for data manipulation and machine learning such as NumPy and sci-kit learn.

* NLP Tasks

A. Tokenization - splitting the text into individual words or subwords, i.e tokens

- The words/text can also be split on the basis of white spaces or punctuation

void -
- The tokenize function of NLTK is used.

Example

```
impat nltk
text = "My name is Pooja"
tokens = nltk.word_tokenize(text)
```

The tokens could be: ['My', 'name', 'is', 'Pooja']

B. Lowercasing

→ convert all text to lowercase to make it case-insensitive

→ use the lower() method

from (A)

eg. Lowercase_tokens = ~~tokens~~ tokens.lower()

c. Remove punctuation

→ Removing punctuation simplifies the text and makes it easier to process.

→ The string module can be used to check if each token is a punctuation character.

eg. filtered_tokens = [token for token in tokens if token not in string.punctuation]

D. Removing stop words

→ Removing words that don't add significant meaning to the text, such as 'a', 'an', 'the'

→ The nltk.corpus.stopwords.words() can be used to get a list of stopwords in a specific language, and the tokens can be filtered from this list

eg. stopwords = nltk.corpus.stopwords.words('english)

removed_stopwords = [token for token in tokens if token.lower() not in stopwords]

5. Removing extra white spaces

→ White spaces could mean spaces, tabs, and newlines that don't add value to further analysis.

→ string.strip() can be used to remove leading and trailing white spaces.

→ string.replace / join can be used to replace multiple consecutive white space characters with a single space.

```
text = text.strip()
text = " ".join(text.split())
```

F. Remove URLs and HTML code

→ A regular expression pattern can be used to identify URLs and replace them with an empty.

for URLs

```
pattern = r"(http|ftp|https)://([\w-]+....)"
cleaned_text = re.sub(pattern, "", text)
```

for HTML

```
pattern = r"<[^>]+>"
cleaned_text = re.sub(pattern, "", text)
```

6. Filter out similar words

→ use nltk.FreqDist() can be used to calculate the frequency of each word and filter out the most common words.

```
text = ''                                    ''

tokens = nltk.word_tokenize(text)

fdist = nltk.FreqDist(tokens)
```

If you want to remove the top 10% of the most commonly occurring words

```
filtered_tokens = [token for token in tokens if fdist[token] <
                                              fdist.N() * 0.1]
```

H. Spelling correction

→ Correcting misspelled words is important to interpret the meaning of a sentence.

→ nltk.corpus.words.words() can be used to get a list of English words

→ nltk.edit_distance() can be used to calculate the edit distance between a word and the words in a list.

```
words = nltk.corpus.words.words()

corrected_tokens[ ]

for token in tokens:

    corrected_word = min(words, key = lambda x: nltk.edit_
                                    distance(x, token))

    corrected_tokens.append(corrected_word)
```

I. Stemming

→ Stemming involves reducing words to their base form, such as jumping to jump.

→ nltk.stem.PorterStemmer() can be used to create a stemmer object

→ The stem() function can be used to remove common morphological affixes from words, to obtain their root

eg.

```
stemmer = nltk.stem.PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in tokens]
```

J. Lemmatization

→ It is a more accurate method of reducing words to their base form than stemming.

→ nltk.stem.WordNetLemmatizer() can be used to create a lemmatizer object.

→ The lemmatize() function can be used for lemmatization.

→ The WordNetLemmatizer uses the WordNet database of English words to lemmatize the tokens, taking into account the part of speech and the context in which the word is used.

→ The part of speech of the token can be specified using the pos argument of the lemmatize() method.

```
lemmatizer = nltk.stem.WordNetLemmatizer()

lemmatized_tokens = [lemmatizer.lemmatize(token) for token
                                                  in tokens]
```

K. Part-of-Speech Tagging

→ used to identify the part of speech of each word in the text, such as noun, verb or adjective.

→ nltk.pos_tag() can be used to tag tokens with their POS tags

eg.

```
tagged_tokens = nltk.pos_tag(tokens)
```

L. Named Entity Recognition

→ involves extracting named entities from a text, like a person's name

→ nltk.ne_chunk() can be used to identify and label named entities in the tokens.

```
eg. tagged_tokens = nltk.pos_tag(tokens)

named_entities = nltk.ne_chunk(tagged_tokens)
```

M. Normalization

→ refers to standardizing words or phrases that have multiple possible forms or spellings.

→ can be done with a list of synonyms or industry-specific terms.

* Information Retrieval

→ involves enabling efficient and effective access to relevant information from a vast collection of documents.

→ can be used for text-classification, where IR methods are used to identify and extract documents that belong to a particular category or class.

→ IR techniques can also be used in sentiment analysis. IR techniques can be used to retrieve reviews that are either positive or negative in sentiment. The retrieved documents provide valuable training data for the classification model, enabling it to learn the characteristics that distinguish between different categories.

→ IR techniques are also used in NLP tasks like question answer - where algorithms are used to retrieve documents that contain the information necessary to answer a user's question.

* Text Classification Models for Spam Detection

→ Consider the training set as follows:

spam messages - spam examples already in spam folder

ham - regular examples in main inbox

① N-gram models

→ Define one n-gram model for P(message|spam) by training on the spam folder, and one model for P(message | ham) by training on the inbox

→ Apply bayes rule: $\text{argmax}_{c \in \{spam, ham\}} P(c|msg) = \text{argmax}_{c \in \{spam, ham\}} P(msg|c).P(c)$

P(c) is estimated by counting the total number of spam and ham messages.

→ The message is represented as a series of feature-value pairs, and apply a classification algorithm to the feature X.

→ If there are 100,000 words in the language model, then the feature vector has length 100,000, but for a short email, almost all the features will have a count of zero. This unigram representation is called the bag of words model.

→ Features have to be constantly updated, because spam detection is an adversarial task, the spammer modifies their spam in response to the spam detector's changes.

→ It is expensive to run algorithms to run algorithms on a very large feature vector, so the process of feature selection is used to keep the best discriminating features between spam and ham.

② Classification by data compression

→ A lossless compression algorithm takes a sequence of symbols, detects repeated patterns in it, and writes a description of the sequence that is more compact than the original.

→ The LZW algorithm models a maximum entropy probability distribution

→ To do classification by compression, lump together all of the spam/ham training messages and compress them as a unit

→ When given a new message to classify, we append it to the

spam & ham messages and compress it.

→ Whichever class compresses better, ie adds a fewer number of additional bytes for the new message is the predicted class.

* Evaluation of Models

1. With cross-validation

→ split the corpus into training & validation corpus

→ determine parameters of the model from the training data, evaluate the model on the validation corpus.

2. Task specific metrics

→ measure accuracy on language identification

→ can also use a task-independent model of language quality

3. Perplexity

→ measuring the probability of a sequence

given by:

$$\text{perplexity}(c_{1:n}) = P(c_{1:n})^{-1/n}$$

* Robotics

Robots and their types

Robots - physical agents that are able to perform tasks by manipulating the physical world.

Types of Robots

(i) Manipulators - industrial robots, assist surgeons in hospitals

(ii) Mobile robots - delivering food in hospital, moving containers at loading docks, UAV, planetary rovers

(iii) Mobile manipulators - humanoid robots

(iv) Intelligent environments - houses equipped w/ sensors and effectors

(v) Multibody systems - robotic action is achieved through swarms of small cooperating robots

* Robot Hardware

① Sensors

(i) Passive Sensors - true observers of the environment, they capture signals that are generated by other sources in the environment

(ii) Active Sensors - such as sonars, send energy into the environment

Active sensors provide more information than passive sensors, but at the expense of increased power consumption, and with a danger of interference when multiple active sensors are used at the same time.

② Range Finders - measure the distance to nearby objects

③ Sonar sensors - emit directional sound waves, which are reflected by objects, with some sound making it back into the sensor.

(4) Stereo Vision - relies on multiple cameras to image the environment from slightly different viewpoints, analyzing the resulting parallax in these images to compute the range of surrounding objects

(5) Scanning lidars - (light detection and ranging) provide longer ranges than time of flight cameras, and tend to perform better in bright daylight

(6) Radar sensors - can measure distances of multiple kilometers

(7) Tactile sensors - measure range based on physical contact, and can be deployed only for sensing objects very close to the robot.

(8) GPS - measures the distance to satellites that emit pulsed signals

(9) Proprioceptive sensors - inform robots of its own motion

shaft decoders - count the revolution of motors in small increments

odometry - measure the distance travelled

(10) Inertial sensors - like gyroscopes - rely on the resistance of mass to the change of velocity

(11) Force and Torque sensors - when robots handle fragile objects or objects whose exact shape and location is unknown.

* Effectors

→ The means by which robots move and change the shape of their bodies.

→ The degree of freedom is counted as each independent direction in which one of the robot's effectors can move.

(i) Differential vs. Synchro Drive

differential - possess two independently actuated wheels, one on each side. If both wheels move at the same velocity, the robot moves on a straight line, if they move in opposite directions, the robot turns on the spot.

synchro - each wheel can move and turn around on its own axis

(ii) Dynamically vs. Statically Stable

dynamically stable - can remain upright while hopping around

statically stable - can remain upright without moving its legs.

(iii) Actuation mechanisms

A. The electric motor for manipulator actuation & locomotion

B. pneumatic actuation using compressed gas

C. hydraulic actuation using pressurized fluids

* Robotic Perception

→ Perception is the process by which robots map sensor measurements into internal representations of the environment

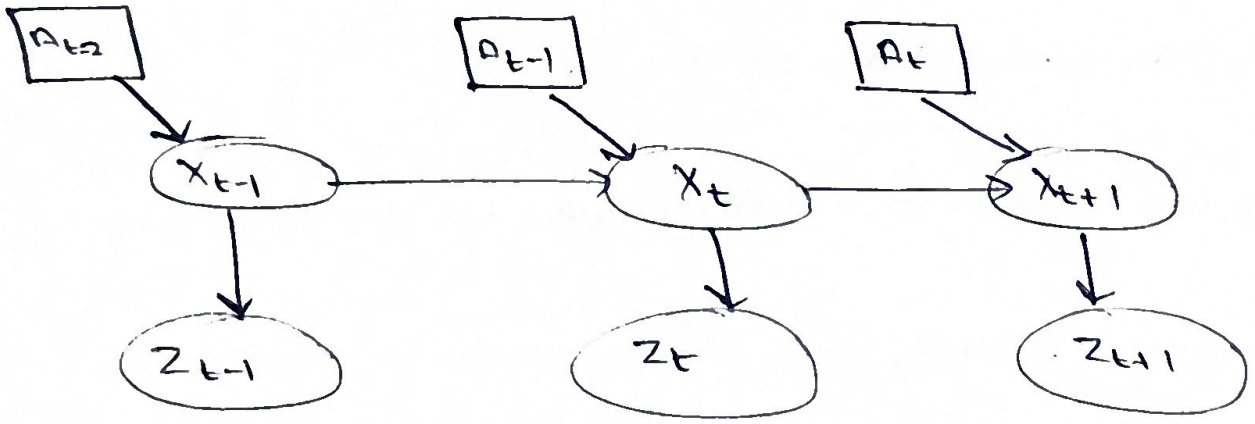
→ Robots have the problems of state estimation.

→ Good internal representations for robots have 3 properties:

(i) contain enough info for the robot to make good decisions

(ii) they are structured, so that they can be updated efficiently

(iii) the internal variables correspond to natural state variables in the physical world.



X_t is the state of the environment, including the robot at time t, Z_t is the observation received at time t, and A_t is the action taken after the observation is received.

$$P(X_{t+1} \mid Z_{1:t+1}, a_{1:t}) =$$
$$\alpha\, P(z_{t+1} \mid X_{t+1}) \int P(X_{t+1} \mid X_t, a_t)\, P(X_t \mid z_{1:t}, a_{1:t-1})\, dX_t$$

→ The state variables at time t+1 is calculated recursively from the corresponding estimate one time step earlier

$P(X_{t+1} \mid x_t, a_t)$ = transition model or motion model

$P(z_{t+1} \mid X_{t+1})$ = sensor model

eg. X_{t+1} might be location of a ball relative to the robot.

* Localization and Mapping

→ Localization is the problem of finding out where things are - including the robot itself.

→ To find out location of objects / navigating robots must know their way around.

→ Consider a robot moving in a flat 2D world.

Its position / state is given by: $X_t = (x_t, y_t, \theta_t)^T$

→ Each action consists of the instantaneous specification of 2 velocities a translational velocity v_t and a rotational velocity ω_t

→ A deterministic model of such motion is given by:

$$\hat{X}_{t+1} = f(X_t, v_t, \omega_t) = X_t + \begin{bmatrix} v_t \Delta t \cos\theta_t \\ v_t \Delta t \sin\theta_t \\ \omega_t \Delta t \end{bmatrix}$$

* Landmarks

→ sensors that detect stable, recognizable features of the environment are called landmarks

→ For each landmark, the range & bearing are reported

$$\hat{z}_t = h(x_t) = \begin{pmatrix} \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \\ \arctan\left(\frac{y_t - y_i}{x_t - x_i}\right) - \theta_t \end{pmatrix}$$

→ Localization using particle filtering is called Monte Carlo Localization

* Information Retrieval Methods

(i) Boolean - AND, NOT, OR } structured queries

(ii) Vector Space Model (with Cosine similarity)

→ represents documents and queries as vectors in a multi-dimensional space, where each term is a dimension

→ cosine similarity is used to measure the similarity between the query vector and document vectors

Vector Representation - document & query is represented as a vector of term wts. weighting schemes like TF-IDF are used

cosine similarity measures the angle between 2 vectors (-1 to 1)

(iii) Inverted Index Method w/ Term Frequency

→ maintain a list of documents for each term
→ provides fast access to documents w/ a specific term

Term frequency - no. of times a term appears in doc - relative importance.

IR methods
boolean IR
vectorspace - cosine similarity
inverted index - term frequency

parsing
NLG
IR and IE
structured
planning - contour algorithm
route finding problem solving

* Parsing in NLP

→ refers to the process of analyzing the grammatical structure of a sentence to understand its components and relationships

→ The primary goal of parsing is to extract meaningful information from text by identifying the syntactic structure of the language

→ It involves breaking down sentences into its constituent parts like nouns, verbs, adjectives & determining how they relate to each other in terms of grammar structure.

→ There are 2 types of parsing:

(i) syntactic parsing - id grammatical relationships

Make a dependency tree, where nodes represent words

(ii) semantic parsing - try to understand meaning of a sentence.

* NLG vs. NLU

↓

generate NL from non-linguistic data, typically structured info/data

convert data into coherent human language

→ used in report generation, summarization, chatbots

→ extract meaning from NL input

comprehension of text or speech by a computer system to understand the user's intent, entities mentioned & context

→ used in sentiment analysis & language translation

IR vs. IE → automatically extracting structured info / knowledge from unstructured text

↓

obtain info relevant to an info need from a large repository or dataset, search & retrieve documents as per request

doesn't retrieve the whole doc, only specific entities, relationships and events