- A matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed from a large piece of text.
- A mathematical technique called Singular Value Decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns.
- Documents are then compared by cosine similarity between any two columns.
  - Values close to 1 represent very similar documents
  - Values close to 0 represent very dissimilar documents.
- The SVD is typically computed using large matrix methods and computed incrementally.
- It uses a neural network-like approach to compress resources via a neural network-like approach. As a result, it does not require the large, full-rank matrix to be held in memory.
- LSA has been used to assist in performing prior art searches for patents.

LSA Process

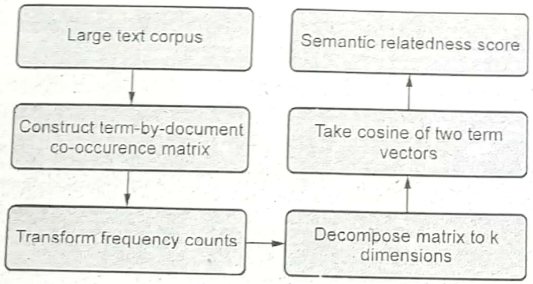


Fig. 2.8.1

Limitations of LSA

- Due to size of the resulting dimensions, it is difficult to interpret.
- LSA can only partially capture polysemy (i.e., multiple meanings of a word) because each occurrence of a word is treated as having the same meaning due to the word being represented as a single point in space.
- The probabilistic model of LSA does not match observed data : LSA assumes that words and documents form a joint Gaussian model (ergodic hypothesis), while a Poisson distribution has been observed.

Review Question

1. *What is latent semantic analysis ?*

❑❑❑

Unit III

3 Language Modelling

Syllabus

Probabilistic language modeling, Markov models, Generative models of language, Log-Liner Models, Graph-based Models
N-gram models : *Simple n-gram models, Estimation parameters and smoothing, Evaluating language models,* ***Word Embeddings / Vector Semantics :*** *Bag-of-words, TFIDF, word2vec, doc2vec, Contextualized representations (BERT)*
Topic Modelling : *Latent Dirichlet Allocation (LDA), Latent Semantic Analysis, Non Negative Matrix Factorization*

Contents

3.1 Introduction

- Natural language processing is different than that of formal languages or programming languages. This is primarily because.
  - Formal languages / programming languages, can be fully specified.
  - All the reserved words in formal language can be defined and identified vice a versa.
- However, it is not possible with natural language. Natural languages are not designed; they evolve continually and therefore there is no formal specification.
- In case of formal languages, there are formal rules for parts of the language and heuristics, but natural language that does not confirm is often used.
- Natural languages involve vast numbers of terms that do not create ambiguities for human brains but create ambiguities in understanding by means of computers.
- An alternative approach is to specify the model of the natural language.
- The key knowledge of the almost 70 years of research in natural language processing can be captured through the use of a small number of formal models or theories.
- Language models are a crucial component in the Natural Language Processing (NLP).
- These language models power all the popular NLP applications we are familiar with - Google Assistant, Siri, Amazon's Alexa, etc.

Language modeling for NLP

- Statistical Language Modeling or Language Modeling and LM for short, is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede it.
- A language model learns the probability of word occurrence based on examples of text. Simpler models may look at a context of a short sequence of words, whereas larger models may work at the level of sentences or paragraphs. Most commonly, language models operate at the level of words.
- A language model can be developed and used standalone, such as to generate new sequences of text that appear to have come from the corpus.
- Language modeling is a root problem for a large range of natural language processing tasks. More practically, language models are used on the front-end or back-end of a more sophisticated model for a task that requires language understanding.
- Language modeling is central to many important natural language processing tasks.
- Recently, neural-network-based language models have demonstrated better performance than classical methods both standalone and as part of more challenging natural language processing tasks.

- There may be formal rules for parts of the language, and heuristics, but natural language that does not confirm is often used. Natural languages involve vast numbers of terms that can be used in ways that introduce all kinds of ambiguities, yet can still be understood by other humans.
- Further, languages change, word usages change: it is a moving target. Nevertheless, linguists try to specify the language with formal grammars and structures. It can be done, but it is very difficult and the results can be fragile.
- An alternative approach to specifying the model of the language is to learn it from examples.
- There are primarily two types of language models :
  1. **Statistical language models :** These models use traditional statistical techniques like N-grams, Hidden Markov Models (HMM) and certain linguistic rules to learn the probability distribution of words.
  2. **Neural language models :** These are new in the NLP town and have surpassed the statistical language models in their effectiveness. They use different kinds of neural networks to model language.
- These models and theories are all drawn from the standard toolkits of Computer Science, Mathematics and Linguistics and should be generally familiar to the experts in the respective fields. One can re-visit these terms (highlighted below) from the course Theory of Computation in the earlier years of Computer Science/ Computer Engineering.

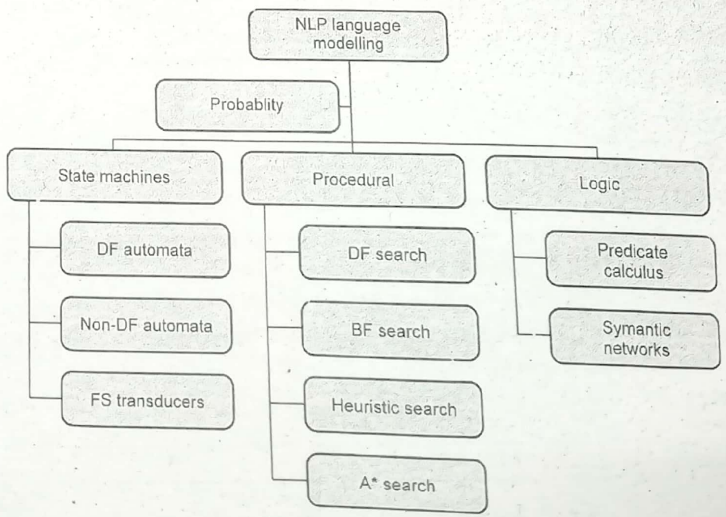


Fig. 3.1.1 Spectrum of NLP language modeling

Statistical language model

- A statistical language model is a probability distribution over sequences of words.
- Given such a sequence, say of length m, it assigns $P(w_1, \ldots, w_m)$ a to the whole sequence.
- The language model provides context to distinguish between words and phrases that sound phonetically similar.
  For example, in English, the phrases "do raid" and "do red" sound similar, but mean different things.
- Data sparsity is a major problem in building language models. This is primarily due to limitations in data modelling. Most possible word sequences are not observed in training.
- One solution is to make the assumption that the probability of a word only depends on the previous n words. This is known as an n-gram model or unigram model when n = 1.
- The unigram model is also known as the bag of words model.
- Estimating the relative likelihood of different phrases is useful in many natural language processing applications, especially those that generate text as an output.
- Discussed as in above example, of "do raid" and "do red", sounds are matched with word sequences.
- Ambiguities are easier to resolve when evidence from the language model is integrated with a pronunciation model and an acoustic model.
- Language models are used in information retrieval in the query likelihood model. There, a separate language model is associated with each document in a collection.
- Documents are ranked based on the probability of the query Q in the document's language model M_d : $P(Q \mid M_d)$. Commonly, the unigram language model is used for document ranking.

Review Questions

1. *What is language modeling ?*
2. *Explain types of language models.*

3.2 Probabilistic Language Modeling

- Probabilistic language models 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.

- Later these models apply these rules in language tasks to accurately predict or produce new sentences.
- The model essentially learns the features and characteristics of basic language and uses those features to understand new phrases.
- A popular idea in computational linguistics is to create a probabilistic model of language.
- Such a model assigns a probability to every sentence in English in such a way that more likely sentences (in some sense) get higher probability.
- In case of the need of disambiguation between two possible sentences, the model picks the sentence with the higher probability.
- Probabilistic language models are distributions over sentence
- There are several different probabilistic approaches to modeling language, which vary depending on the purpose of the language model.
- From a technical perspective, the various types differ by the amount of text data they analyze and the algorithm they use to analyze it.
- Some popular probabilistic language models are :
  1. Unigram : Words generated one at a time, drawn from a fixed distribution.
  2. Bigram : Probability of word depends on previous word.
  3. Tag bigram : Probability of part of speech depends on previous part of speech, probability of word depends on part of speech.
  4. Maximum entropy (Exponential) : The model evaluates text using an equation that combines feature functions and n-grams.
  5. Continuous space : This type of model represents words as a non-linear combination of weights in a neural network.
- Any complex probabilistic language models is recommended for NLP tasks, because natural language itself is extremely complex and always evolving.
- As a result, an exponential model or continuous space model might be better than an n-gram for NLP tasks, because they are designed to account for ambiguity and variation in language.

Review Question

1. *Explain probabilistic language modeling.*

3.3 Markov Models

- Markov model is an un-precised model that is used in the systems that does not have any fixed patterns of occurrence i.e., randomly changing systems.

- Markov model is based upon the fact of having a random probability distribution or pattern that may be analyzed statistically but cannot be predicted precisely.
- In Markov model, it is assumed that the future states only depend upon the current states and not the previously occurred states.
- There are four common Markov models out of which the most commonly used is the hidden Markov model.

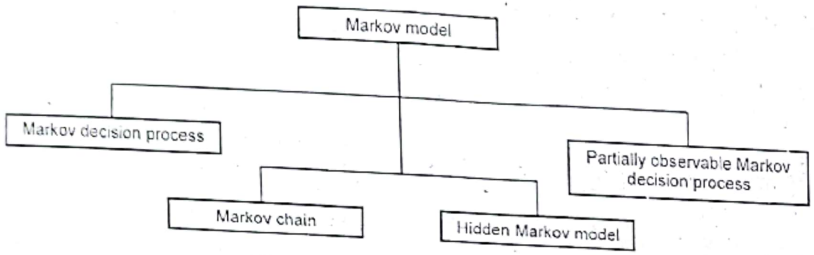


Fig. 3.3.1 Types of Markov models

- For NLP, a Markov chain can be used to generate a sequence of words that form a complete sentence, or a hidden Markov model can be used for named-entity recognition and tagging parts of speech.
- For machine learning, Markov decision processes are used to represent reward in reinforcement learning.
- The Hidden Markov model (popularly known HMM) is a probabilistic model which is used to explain or derive the probabilistic characteristic of any random process.
- It basically says that an observed event will not be corresponding to its step-by-step status but related to a set of probability distributions.

Markov model of natural language

- Claude Shannon approximated the statistical structure of a piece of text using a simple mathematical model known as a Markov model.
- A Markov model of order 0 predicts that each letter in the alphabet occurs with a fixed probability.
- We can fit a Markov model of order 0 to a specific piece of text by counting the number of occurrences of each letter in that text and using these counts as probabilities.

- For example, if the input text is "agggcagcgggcg", then the Markov model of order 0 predicts that each letter is 'a' with probability 2/13, 'c' with probability 3/13 and 'g' with probability 8/13.
- The following sequence of letters is a typical example generated from this model.

```
a g g c g a g g g a g c g g c a g g g g...
```

- An order 0 model assumes that each letter is chosen independently. This does not coincide with the statistical properties of English text since there is a high correlation among successive letters in an English word or sentence. For example, the letters 'h' and 'r' are much more likely to follow 't' than either 'c' or 'x'.
- We obtain a more refined model by allowing the probability of choosing each successive letter to depend on the preceding letter or letters.
- A Markov model of order k predicts that each letter occurs with a fixed probability, but that probability can depend on the previous k consecutive letters (k-gram)

Hidden Markov Model

- In Hidden Markov Model, every individual states has limited number of transitions and emissions.
- Probability is assigned for each transition between states. Hence, the past states are totally independent of future states.
- The fact that HMM is called hidden because of its ability of being a memory less process i.e. its future and past states are not dependent on each other.
- Since HMM is rich in mathematical structure it can be implemented for practical applications. This can be achieved on two algorithms called as :
  - Forward algorithm.
  - Backward algorithm.
- The main goal of HMM is to learn about a Markov chain by observing its hidden states.
- Considering a Markov process X with hidden states Y here the HMM solidifies that for each time stamp the probability distribution of Y must not depend on the history of X according to that time.
- Refer to the Hidden Markov chain in below figure, that indicates the N states with probability indicated by the weight of transition arch.

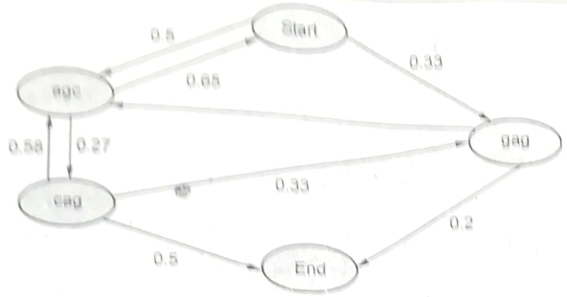


Fig. 3.3.2 Example of Hidden Markov chain for NLP application

- The applications where the HMM can be used have sequential data like time series data, audio and video data and text data or NLP data.
- A prominent NLP application of the HMM is in the Part-of-Speech tagging.

Review Questions

1. *What are Markov models.*
2. *Explain Markov model of natural languages.*
3. *Explain Hidden Markov Model.*

3.4 Generative Models for Language

- The generative model is a single platform for diversified areas of NLP that can address specific problems relating to read text, hear speech, interpret it, measure sentiment and determine which parts are important.
- This is achieved by process of elimination once the relevant components are identified. Single platform provides same model generating and reproducing optimized solutions and addressing different issues.
- Generative models are one of the most promising approaches towards this goal. To train a generative model we first collect a large amount of data in some domain (e.g., think millions of images, sentences, or sounds, etc.) and then train a model to generate data like it. The intuition behind this approach follows a famous quote from Richard Feynman.
- Generative models have many short - term applications. But in the long run, they hold the potential to automatically learn the natural features of a dataset, whether categories or dimensions or something else entirely.

- Mathematically, we think about a dataset of examples x1, x2 ,x3, ... xn as samples from a true data distribution p(x).
- In the example image below, the region shows the part of the image space that, with a high probability (over some threshold) contains real images and black dots indicate our data points (each is one image in our dataset).
- Now, the model also describes a distribution $\hat{p}(x)$ that is defined implicitly by taking points from a unit Gaussian distribution and mapping them through a (deterministic) neural network-our generative model.

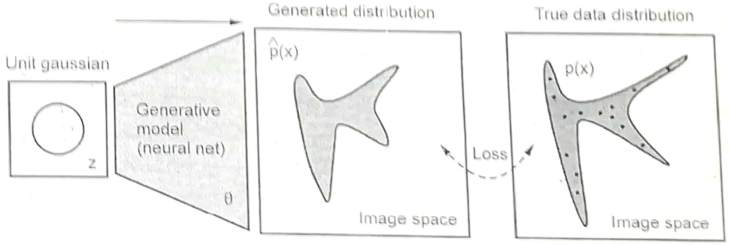


Fig. 3.4.1 Generative model example

Three approaches to generative models

Most generative models have this basic setup, but differ in the details. Here are three popular examples of generative model approaches to give you a sense of the variation :

- **Generative Adversarial Networks (GANs)**
  - This is already discussed above, pose the training process as a game between two separate networks : A generator network (as seen above) and a second discriminative network that tries to classify samples as either coming from the true distribution p(x) or the model distribution $\hat{p}(x)$.
  - Every time the discriminator notices a difference between the two distributions the generator adjusts its parameters slightly to make it go away, until at the end (in theory) the generator exactly reproduces the true data distribution and the discriminator is guessing at random, unable to find a difference.
- **Variational Autoencoders (VAEs)**
  - It allow us to formalize this problem in the framework of probabilistic graphical models where we are maximizing a lower bound on the log likelihood of the data.

- **Autoregressive models**
  - Models such as PixelRNN instead train a network that models the conditional distribution of every individual pixel given previous pixels (to the left and to the top).
  - This is similar to plugging the pixels of the image into a char-rnn, but the RNNs run both horizontally and vertically over the image instead of just a 1D sequence of characters.

Review Question

1. *Explain generative models language.*

3.5 Log Linear Model

- Log-linear models, which are very widely used in natural language processing. A key advantage of log-linear models is their flexibility
- Log-linear models have become a widely-used tool in NLP classification tasks.
- A log-linear model consists of the following components :
  - A set X of possible inputs.
  - A set Y of possible labels. The set Y is assumed to be finite.
  - A positive integer d specifying the number of features and parameters in the model.
  - A function $f : X \times Y \rightarrow R^d$ that maps any (x, y) pair to a feature-vector f(x, y).
  - A parameter vector $v \in R^d$.
- Log-linear models assign joint probabilities to observation/label pairs $(x, y) \in X \times Y$ as follows :

$$P_{\vec{\theta}}(x, y) = \frac{\exp\left(\vec{\theta} \cdot \vec{f}(x, y)\right)}{\Sigma_{x',y'} \exp\left(\vec{\theta} \cdot \vec{f}(x, y)\right)}$$

where ~θ is a R-valued vector of feature weights

~ f is a function that maps pairs (x, y) to a nonnegative R-valued feature vector.

- These features can take on any form; in particular, unlike directed, generative models (like HMMs and PCFGs), the features may overlap, predicting parts of the data more than once.
- Each feature has an associated θ_i, which is called its weight. Maximum likelihood parameter estimation (training) for such a model, with a set of labeled examples, amounts to solving the following optimization problem. Let $\{(x_1, y*1), (x_2, y*2), \ldots, (x_m, y*m)\}$.

Review Question

1. *Explain log linear model.*

3.6 N-Gram Model, Simple N Gram Models

- To understand concept of n gram models consider the example sentence.
- Please turn your homework .....
- We want to predict the next word which can be 'in', 'over' but definitely it will not the word 'the'. This is known as word prediction and can be done using probabilistic models called as n-gram models.
- In n-gram model from sequence of n-token words the next word is predicted from previous n - 1 words.
- It is difficult to compute probability of any word sequence w.
- It can be computed by decomposing it based on chain rule of probality as :

$$P(w) = P(w_1 \ldots w_t) = (P)(w_1) \prod_{i=1}^{+} P(w_i / w_{i-1} w_{i-2} \ldots w_2 w_1)$$

- But as individual product terms also cannot be computed directly n-gram approximation will be useful.
- By considering the assumption of history equivalence class that n–1 are useful for predicting a given word, n-gram model can be defined as :

$$P(w) \approx \prod_{i=1}^{+} P(w_i | w_{i-1}, \ldots w_{i-n+1})$$

- This is based on markov assumption that current word only depends on n – 1 preceding words and in dependent of all the other given words.
- So n-gram model is also called as $(n-1)^{-th}$ order Markov model.
- Based on length of n the models can be formalized as : for n = 2 → biagram → considering two word sequence of words ex. "please turn" or "turn your" for n=3 → trigram → considering three word sequence of words ex: "please turn your" or "turn your homework" and so on for n = 4,5, etc.

Review Question

1. *Explain N Gram model.*

3.7 Estimation Parameter and Smoothing

- Generally n-gram probabilities are estimated by combining maximum likelihood criterion with parameter smoothing.
- The maximum likelihood estimate can be obtained as

$$P(w_i \mid w_{i-1}, w_{i-2}) = \frac{C(w_i, w_{i-1}, w_{i-2})}{C(w_{i-1}, w_{i-2})}$$

Where $C(w_i, w_{i-1}, w_{i-2})$ is count of trigram w_{i-2}, w_{i-1}, w_i in training data.

- Smoothing is the process of flattening the peakes in n-gram probability distribution by redistributing probability mass. Also zero estimates are replaced by some small nonzero values.
- One of the common smoothing technique is called as back off.
- It splits n-grams whose count in training data fall below predetermined threshold T and also whose count exceed the threshold.
- The backed off probability P_{BO} for w_i given w_{i-1}, w_{i-2} is computed as

$$P_{BO}(w_i \mid w_{i-1}, w_{i-2}) = \begin{cases} d_c P(w_i \mid w_{i-1}, w_{i-2}) \text{ it } C > T \\ d(w_{i-1}, w_{i-2}) P_{BO}(w_i \mid w_{i-1}) \text{ otherwise} \end{cases}$$

Where $C \rightarrow$ count of (w_i, w_{i-1}, w_{i-2})

$d_c \rightarrow$ discounting factor applied to higher order distribution.

- It is a parameter estimation method in which set of parameters of a model are considered as random variable, which is governed by previous statistical distribution posterior distribution,
- $P(\theta \mid S)$ is given using Bay's rule as

$$P(\theta \mid S) = \frac{P(S \mid \theta) P(\theta)}{P(s)}$$

Where $S \rightarrow$ training sample with sequence of words $w_1 \ldots\ldots w_t < P(w_i), \ldots,$

$P(w_k) >$ (where k is vocabulary size)

$\theta \rightarrow$ Set of parameters

= for unigram model $P(w_1 \mid h_1 \ldots\ldots, P(w_k \mid h_k)) >$ for n-gram

$P(\theta) \rightarrow$ Prior distribution over different possible values of θ

- A point estimate of θ is done by Maximum a Posteriori (MAP) criteria as follows.

$$\theta_{MAP} = \underset{\theta \in \Theta}{\text{argmax}}\, P(\theta \mid s) = \underset{\theta \in \Theta}{\text{argmax}}\, P(S \mid \theta) P(\theta)$$

Where Θ is space for possible assignments for θ

- The expected value of θ for sample S is given as :

$$\theta_B = E[\theta \mid S] = \int_{\Theta} \theta P(\theta \mid S) d\theta$$

$$= \frac{\int_{\Theta} \theta P(S \mid \theta) P(\theta) d\theta}{\int_{\Theta} P(S \mid \theta) P(\theta) d\theta}$$

- With the increase of monolingual data scaling of language is required to handle sets of bilions or trillions of words.
- In this case exact probability computations and in turn parameter estimation is not feasible.
- So to handle the large data, complete data required for training is divided into partitiones.
- Probabilities derived from each partition are stored in separate physical location.
- The language model server handles this data which is distributed over a cluster of independent nodes.
- Clients request statistics from this server during runtime.
- These models facilitate scalabity for handling large amount of data, also new data can be added dynamically.
- The disadvantage is slow speed due to networking overheads.
- Another variation can be use of large scale. Distributed language at second pass rescoring stage.
- In this first pass hypothesis is done using smaller language models.
- One more approach is storing large scale, language models in working memory of a single machine.
- The concept of bloom filter is used for this purpose.
- The corpus statistics is stored in memory efficient and randomized data structure called as Bloom filter in quantized manner.

Review Questions

1. *Explain the concept of estimation parameter and smoothing.*
2. *Explain large scale languages models.*

3.8 Evaluating Language Model

- Evaluating the performance of language model is the best way to include it in an application and check the performance of that application. This process is called as extrinsic evaluation.
- But as it is expensive at times, another way of evolution i.e. intrinsic evolution is used.
- In intrinsic evolution a metric is used to quickly evaluate the improvements in language model.
- Two criterias used for intrinsic evaluation of language model are.

1. **Coverage rate**
   - It measures percentage of n-grams in test set
   - Sometimes there are cases where same unknown words appear they are called as Out Of Vocabulary (OOV) words which cannot be handled by this type.
2. **Perplexity**
   - It considers the fact that among two probabilistic models the model which fits the test data is the better one.

Review Question

1. *How to evaluate language model ?*

3.9 Bag of Words

- In NLP the input to various algorithms for different applications is in text form.
- But these machine learning algorithms cannot work on the raw text.
- The text must be converted to numbers for further processing.
- More specifically the vectors of these numbers representing text are generated.
- Bag Of Words (BOW) representation is used to convert text into fixed length vectors.
- BOW moded takes into account the frequency of occurrence of words in input text document.
- It focuses on two things :
  1. Vocabulary of known words.
  2. Measure of presence of these known words.

- The word 'Bag' in bag of words is analogous to the bag of items. When we fill a bag the order of the items is discarded, similarly in bag of words model represents unordered set of words irrespective of their position in the document.
- It only considers the frequency of words in the text.
- Let's consider the following example to understand how bag of words model works stepwise.

Step 1 : Step 1 includes collection of data. Consider each of the following sentences as text documents.

- The dog sat
- The dog sat in the hat
- The dog with the hat

Step 2 :

- Step 2 includes designing the vocabulary.
- All the words present in document set are listed.
- In out example the words formed are : the, dog, sat, in, the, hat, with.

Step 3 :

- Step 3 includes computation of the frequency of the occurrence of w the document.
- In our example it can be list as :

Document	the	dog	sat	in	hat	with
the dog sat	1	1	1	0	0	0
the dog sat in the hat	2	1	1	1	1	0
the dog with the hat	2	1	0	0	1	1

- With the 6 distinct words and their frequency of occurrence in the docum we can write fixed length vector for each document as follows :
  - The dog sat → [1, 1, 1, 0, 0, 0]
  - The dog sat in the hat → [2, 1, 1, 1, 1, 0]
  - The dog with the hat → [2, 1, 0, 0, 1, 1]
- The drawback of BOW model is, we loose the context of the document.
- BOW model only tells what words will appear in the document frequency but it doesn't tell where they occurred.

- The bag-of-words model is a simplifying representation used in natural language processing and information retrieval.
- In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision.
- The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.
- The Bag-of-words model is one example of a vector space model. Bag of words is a natural language processing technique of text modeling.
- It is a method of feature extraction with text data. This approach is a simple and flexible way of extracting features from documents.
- Whenever we apply any algorithm in NLP, it works on numbers. We cannot directly feed our text into that algorithm. Hence, Bag of Words model is used to preprocess the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently used words.
- This model can be visualized using a table, which contains the count of words corresponding to the word itself.
- The approach is very simple and flexible and can be used in a myriad of ways for extracting features from documents.
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things :
  - A vocabulary of known words.
  - A measure of the presence of known words.
- It is called a "bag" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document
- A bag of words is a representation of text that describes the occurrence of words within a document.
- The model is only concerned with whether known words occur in the document, not where in the document.
- One of the biggest problems with text is that it is messy and unstructured and machine learning algorithms prefer structured, well defined fixed-length inputs and by using the Bag-of-Words technique we can convert variable-length texts into a fixed-length vector.

Steps of bag of words model :

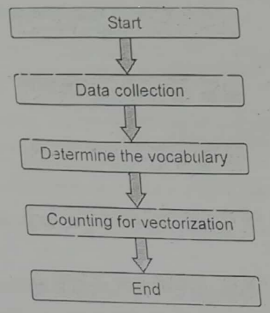


Fig. 3.9.1

Disadvantages of using bag of words model :

- The model ignores the location information of the word. The location information is a piece of very important information in the text
- Bag of word models doesn't respect the semantics of the word. For example, words 'soccer' and 'football' are often used in the same context.
- The range of vocabulary is a big issue faced by the bag-of-words model.

TF / IDF

- TF-IDF stands for Term Frequency Inverse Document Frequency of records.
- It can be defined as the calculation of how relevant a word in a series or corpus is to a text.
- The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

Key concepts in TF / IDF

- Term frequency :
  - In document d, the frequency represents the number of instances of a given word t.
  - Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational.
  - Since the ordering of terms is not significant, we can use a vector to describe the text in the bag of term models.

- For each specific term in the paper, there is an entry with the value being the term frequency.
- The weight of a term that occurs in a document is simply proportional to the term

$$tf(t,d) = \text{count of t in d / number of words in d}$$

- Document frequency :
  - This tests the meaning of the text, which is very similar to TF, in the whole corpus collection.
  - The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t.
  - In other words, the number of papers in which the word is present is DF.

$$df(t) = \text{Occurrence of t in documents}$$

$$tf(t,d) = \text{Count of t in d / number of words in d}$$

- Document frequency :
  - This tests the meaning of the text, which is very similar to TF, in the whole corpus collection.
  - The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t.
  - In other words, the number of papers in which the word is present is DF.

$$df(t) = \text{Occurrence of t in documents}$$

- Inverse document frequency :
  - Mainly, it tests how relevant the word is.
  - The key aim of the search is to locate the appropriate records that fit the demand.
  - Since tf considers all terms equally significant, it is therefore not only possible to use the term frequencies to measure the weight of the term in the paper.
  - First, find the document frequency of a term t by counting the number of documents containing the term :

$$df(t) = N(t)$$

where

$$df(t) = \text{Document frequency of a term t}$$

$$N(t) = \text{Number of documents containing the term t}$$

- Term frequency is the number of instances of a term in a single document only although the frequency of the document is the number of separate documents in which the term appears, it depends on the entire corpus.
- Now let's look at the definition of the frequency of the inverse paper. The IDF of the word is the number of documents in the corpus separated by the frequency of the text.

$$idf(t) = N/\ df(t) = N/N(t)$$

- The more common word is supposed to be considered less significant, but the element (most definite integers) seems too harsh. We then take the logarithm (with base 2) of the inverse frequency of the paper. So the if of the term t becomes :

$$idf(t) = \log(N/\ df(t))$$

- Computation :
  - Tf-idf is one of the best metrics to determine how significant a term is to a text in a series or a corpus.
  - tf-idf is a weighting system that assigns a weight to each word in a document based on its term frequency (tf) and the reciprocal document frequency (tf) (idf).
  - The words with higher scores of weight are deemed to be more significant.
- Usually, the tf-idf weight consists of two terms -
  - Normalized term frequency (tf)
  - Inverse document frequency (idf)
- Hence we get

$$tf\text{-}idf(t, d) = tf(t, d) * idf(t)$$

Applications of TFIDF

- Document classification - Helps in classifying the type and genre of a document.
- Topic modelling - It helps in predicting the topic for a corpus.
- Information retrieval system - To extract the important information out of a corpus.
- Stop word filtering - Helps in removing the unnecessary words out of a text body.

Review Questions

1. *Explain Bag of Words model.*
2. *Explain the concept TF/IDF.*

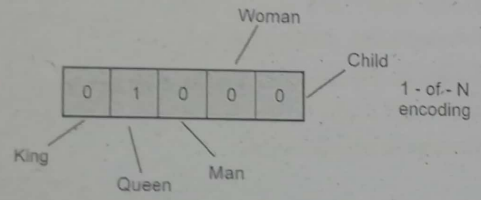
3.10 Word Embedding / Vector Semantics

- As described in the BOW model we have seen how a word in a document can be represented in numerical form as a vector representation.
- In other words this numerical representation of words in called as word embedding.
- We can consider the analogy of RGB representation of colours to understand this concept.
- Section 3.9 illustrates how vector representation can be obtained from sentences in a document.
- Word embedding are classified in two types :
  1. Frequency based embedding
  2. Prediction based embedding

1. Simple frequency based embeddings

One-hot encoding

- Let's start by looking at the simplest way of converting text into a vector.
- We could have a vector of the size of our vocabulary where each element in the vector corresponds to a word from the vocabulary.
- The encoding of a given word would then simply be the vector in which the corresponding element is set to 1 and all other elements are 0.
- Suppose our vocabulary has only five words : King, Queen, Man, Woman and Child. We could encode the word 'Queen' as :



- This simple technique of creating numerical representations is called **One-Hot Encoding**.
- But this is neither scalable and nor smart.
- If we have a vocabulary of ten million words, it would mean that we would need vectors of size 10M, where each word would be represented by a single 1 and all other 9,999,999 elements would be set to 0.

- Besides the sparseness and huge memory requirement issues, this technique is also not able to capture any semantic relationships or context information.
- **Tf-Idf** is another popular technique for computing frequency based embeddings.

2. Prediction based embeddings

- Prediction based embedding is one of the most sought word embedding technique in NLP.
- These methods were prediction-based as they give the probabilities to the words.
- They proved to be state of the art for tasks like word analogies and word similarities.
- They were also able to achieve algebraic operations tasks.
- Following are the two algorithms in prediction based embedding,
  1. Word2Vec
  2. Doc2Vec
- Refer section 3.11 for word2vec model.

3.11 Word2Vec

- Word2vec is a technique for Natural Language Processing (NLP) published in 2013.
- The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.
- Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence.
- As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector.
- The vectors are chosen carefully such that they capture the semantic and syntactic qualities of words; as such, a simple mathematical function (cosine similarity) can indicate the level of semantic similarity between the words represented by those vectors.
- Word2vec can utilize either of two model architectures to produce these distributed representations of words : Continuous bag-of-words (CBOW) or continuous skip-gram.
- In both architectures, word2vec considers both individual words and a sliding window of context words surrounding individual words as it iterates over the entire corpus.

- In the continuous bag-of-words architecture, the model predicts the current word from the window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption).
- In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words.
- The skip-gram architecture weighs nearby context words more heavily than more distant context words. According to the authors' note, CBOW is faster while skip-gram does a better job for infrequent words.
- After the model has trained, the learned word embeddings are positioned in the vector space such that words that share common contexts in the corpus - that is, words that are semantically and syntactically similar - are located close to one another in the space. More dissimilar words are located farther from one another in the space.
- This occurs because words which influence relative word probabilities in similar ways will have learned similar embeddings once the model has finished.

Review Question

1. *Explain word2vec technique.*

3.12 Doc2Vec

- Doc2Vec is an extension to word2vec. doc2vec, generates distributed representations of variable - length pieces of texts, such as sentences, paragraphs, or entire documents.
- Doc2Vec has been implemented in the C, Python and Java/Scala tools, with the Java and Python versions also supporting inference of document embeddings on new, unseen documents.
- Doc2Vec estimates the distributed representations of documents much like how word2vec estimates representations of words.
- Doc2Vec utilizes either of two model architectures, both of which are allegories to the architectures used in word2vec.
  - The first, Distributed Memory Model of Paragraph Vectors (PV-DM), is identical to CBOW other than it also provides a unique document identifier as a piece of additional context.
  - The second architecture, Distributed Bag of Words version of Paragraph Vector (PV-DBOW), is identical to the skip-gram model except that it attempts to predict the window of surrounding context words from the paragraph identifier instead of the current word.

- Doc2Vec also has the ability to capture the semantic 'meanings' for additional pieces of 'context' around words;
- Doc2Vec can estimate the semantic embeddings for speakers or speaker attributes, groups and periods of time.
- For example, doc2vec has been used to estimate the political positions of political parties in various Congresses and Parliaments in the U.S. and U.K., respectively, and various governmental institutions.

3.13 Contextualized Representations (BERT)

- Bidirectional Encoder Representations from Transformers (BERT) is a masked-language model published in 2018 by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
- A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in NLP experiments", counting over 150 research publications analyzing and improving the model.
- BERT was originally implemented in the English language at two model sizes
  - BERTBASE : 12 encoders with 12 bidirectional self-attention heads totaling 110 million parameters,
  - BERTLARGE : 24 encoders with 16 bidirectional self-attention heads totaling 340 million parameters. Both models were pre-trained on the Toronto Books Corpus (800 M words) and English Wikipedia (2,500 M words).
- Unlike previous models, BERT is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus.
- Context-free models such as word2vec or GloVe generate a single word embedding representation for each word in the vocabulary, where BERT takes into account the context for each occurrence of a given word.

BERT architecture

- BERT is based on the transformer architecture. Specifically, BERT is composed of transformer encoder layers.
- BERT was pre-trained simultaneously on two tasks :
  - Language modeling (15 % of tokens were masked and the training objective was to predict the original token given its context).
  - next sentence prediction (the training objective was to classify if two spans of text appeared sequentially in the training corpus).

- As a result of this training process, BERT learns latent representations of words and sentences in context.
- After pre-training, BERT can be fine tuned with fewer resources on smaller datasets to optimize its performance on specific tasks such as
  - NLP tasks (language inference, text classification)
  - Sequence-to-sequence based language generation tasks (question-answering, conversational response generation).
- The pre-training stage is significantly more computationally expensive than fine tuning.
- Researchers are hopeful that BERT will provide a contextualized embedding that will be different according to the sentence.

Review Question

1. *Explain BERT.*

3.14 Topic Modeling

- Topic modeling is recognizing the words from the topics present in the document or the corpus of data.
- This is useful because extracting the words from a document takes more time and is much more complex than extracting them from topics present in the document.
- For example, there are 1000 documents and 600 words in each document. So to process this it requires $600 \times 1000 = 600000$ threads.
- So when you divide the document containing certain topics then if there are 5 topics present in it, the processing is just 5×600 words = 3000 threads.
- This looks simple than processing the entire document and this is how topic modelling has come up to solve the problem and also visualizing things better.
- Some of the important points or topics which makes text processing easier in NLP :
  - Removing stop words and punctuation marks
  - Stemming
  - Lemmatization
  - Encoding them to ML language using count vectorizer or TFIDF vectorizer
- When stemming is applied to the words in the corpus the word gives the base for that particular word. There are different types through which stemming can be performed. Some of the popular ones which are being used are :
  - Porter stemmer
  - Lancaster stemmer
  - Snowball stemmer

- Topic modeling is done using LDA(Latent Dirichlet Allocation). Refer section 3.15.
- Topic modeling refers to the task of identifying topics that best describes a set of documents. These topics will only emerge during the topic modeling process (therefore called latent). One popular topic modeling technique is known as Latent Dirichlet Allocation (LDA).
- Topic modeling is an unsupervised approach of recognizing or extracting the topics by detecting the patterns like clustering algorithms which divides the data into different parts.
- The same happens in Topic modeling in which we get to know the different topics in the document.
- This is done by extracting the patterns of word clusters and frequencies of words in the document. Refer Fig. 3.14.1 to visualize topic modeling

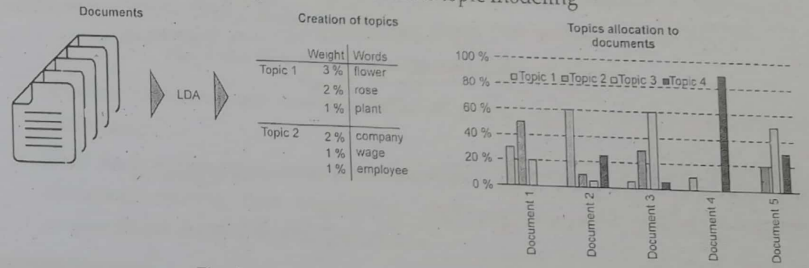


Fig. 3.14.1 Visual representation of topic modeling

Applications of topic modeling :

1. Medical industry
2. Scientific research understanding
3. Investigation reports
4. Recommender system
5. Blockchain
6. Sentiment analysis
7. Text summarisation
8. Query expansion which can be used in search engines

Review Question

1. *Explain topic modeling.*

3.15 Latent Dirichlet Allocation

Bayesian topic based language models or Latent Dirichlet Allocation (LDA) model.

- Bayesian topic based models are emerged recently in statistical language modeling.
- Blei, Ng and Jordan proposed the first LDA model under Bayesian topic based models.
- LDA model is designed on following assumptions :
  - Each document is considered to be formed of k topics and denoted as $Z_1, \ldots$ Z_k.
  - Topic specific distribution over a particular word is used to generate a word $k = 1, \ldots k$. probablity vector ϕ_k is generated.
  - The prior probabilities of each topic (θ_k). $\theta_1, \theta_2, \ldots \theta_k$ are distributed according to Dirichlet distribution with hyper parameters $\alpha_1, \ldots \alpha_k$ is given as

$$P(\theta_1, \ldots \theta_k) = \frac{\Gamma(\sum_k \alpha_k)}{\pi_k \Gamma(\alpha_k)} \prod_{k=1}^{k} \theta_k^{\alpha_k - 1}$$

- Probablity of complete document with sequence W of t words is given as :

$$P(W \mid \alpha, \phi) = \int P(\theta \mid \alpha) \left(\pi_{i=1}^{t} \sum_{z_i} P(z_i \mid \theta) P(w_i \mid z_i, \phi) \right) d\theta$$

Review Question

1. *Explain latent dirichlet allocation.*

3.16 Latent Semantic Analysis

- Latent Semantic Analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.
- LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis).
- A matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed from a large piece of text and a mathematical technique called Singular Value Decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns.

- Documents are then compared by cosine similarity between any two columns.
- Values close to 1 represent very similar documents while values close to 0 represent very dissimilar documents.

Occurrence matrix

- LSA can use a document-term matrix which describes the occurrences of terms in documents; it is a sparse matrix whose rows correspond to terms and whose columns correspond to documents. A typical example of the weighting of the elements of the matrix is tf-idf (term frequency - inverse document frequency) : The weight of an element of the matrix is proportional to the number of times the terms appear in each document, where rare terms are upweighted to reflect their relative importance.

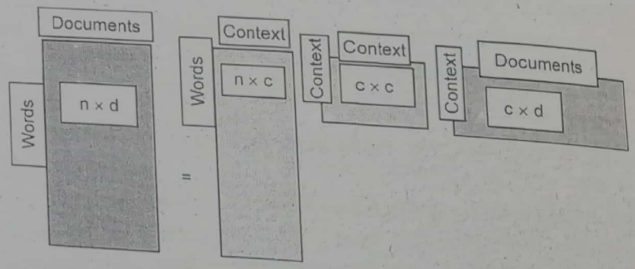


Fig. 3.16.1 LCA occurrence matrix

- This matrix is also common to standard semantic models, though it is not necessarily explicitly expressed as a matrix, since the mathematical properties of matrices are not always used.

Rank lowering

- After the construction of the occurrence matrix, LSA finds a low-rank approximation to the term-document matrix. There could be various reasons for these approximations :
- The original term-document matrix is presumed too large for the computing resources; in this case, the approximated low rank matrix is interpreted as an approximation (a "least and necessary evil").
- The original term-document matrix is presumed noisy : For example, anecdotal instances of terms are to be eliminated. From this point of view, the approximated matrix is interpreted as a de-noisified matrix (a better matrix than the original).

- The original term-document matrix is presumed overly sparse relative to the "true" term-document matrix. That is, the original matrix lists only the words actually in each document, whereas we might be interested in all words related to each document-generally a much larger set due to synonymy.
- This mitigates the problem of identifying synonymy, as the rank lowering is expected to merge the dimensions associated with terms that have similar meanings.
- It also partially mitigates the problem with polysemy, since components of polysemous words that point in the "right" direction are added to the components of words that share a similar meaning.
- Conversely, components that point in other directions tend to either simply cancel out, or, at worst, to be smaller than components in the directions corresponding to the intended sense.

Application of LSA

- Compare the documents in the low-dimensional space (data clustering, document classification).
- Find similar documents across languages, after analyzing a base set of translated documents (cross-language information retrieval).
- Find relations between terms (synonymy and polysemy).
- Given a query of terms, translate it into the low-dimensional space and find matching documents (information retrieval).
- Find the best similarity between small groups of terms, in a semantic way (i.e. in a context of a knowledge corpus), as for example in multiple choice questions MCQ answering model.
- Expand the feature space of machine learning / text mining systems.
- Analyze word association in text corpus.

Review Question

1. *Explain latent semantic analysis.*

3.17 Non Negative Matrix Factorization (NMF)

- Non-Negative Matrix Factorization is a statistical method that helps us to reduce the dimension of the input corpora or corpora.

- Internally, it uses the factor analysis method to give comparatively less weightage to the words that are having less coherence.
- Non-Negative Matrix Factorization (NMF or NNMF), also non-negative matrix approximation is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into (usually) two matrices W and H, with the property that all three matrices have no negative elements.
- This non-negativity makes the resulting matrices easier to inspect. Also, in applications such as processing of audio spectrograms or muscular activity, non-negativity is inherent to the data being considered.

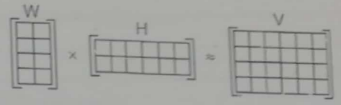


Illustration of approximate non-negative matrix factorization : The matrix **V** is represented by the two smaller matrices **W** and **H**, which, when multiplied, approximately reconstruct **V**.

Fig. 3.17.1 Non-Negative matrix factorization illustrated

- Since the problem is not exactly solvable in general, it is commonly approximated numerically.
- NMF finds applications in such fields as astronomy, computer vision, document clustering, missing data imputation, chemometrics, audio signal processing, recommender systems and bioinformatics.

Review Question

1. *Explain Non-negative matrix factorization.*

❑❑❑

Notes

Unit IV

4 Information Retrieval using NLP

Syllabus

Information Retrieval : *Introduction, Vector Space Model*

Named Entity Recognition : *NER System Building Process, Evaluating NER System Entity Extraction, Relation Extraction, Reference Resolution, Coreference resolution, Cross Lingual Information Retrieval.*

Contents

4.1 Information Retrieval : Introduction

- Information retrieval field mainly contain two broad tasks, first is storage and second is retrieval of all manner of media.
- IR task focuses on storage of text documents and their retrieval according to user's request.
- With IR context, a word in document refer to unit of text in the system, which is indexed and available for retrieval.
- Documents can be of different types for example, newspaper, articles, encyclopedia entries or simply small paragraphs and sentences.
- Some terminologies used in IR systems are,

  collection → Set of documents to satisfy user queries.

  term → Lexical item in collection.

  query → User's information need in set of terms.
- The architecture of Ad hoc IR system is shown in Fig. 4.1.1.

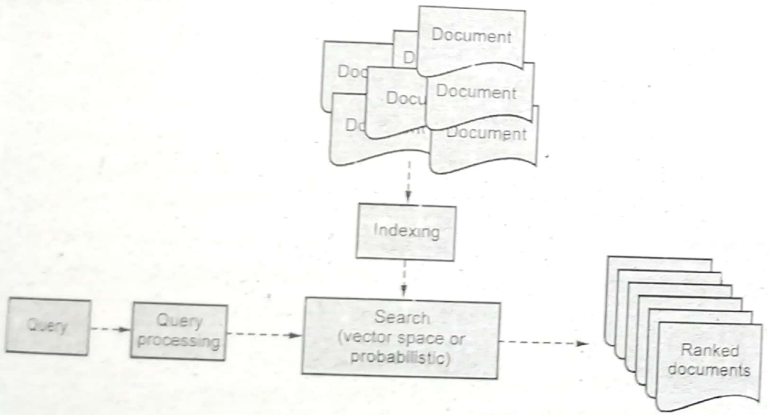


Fig. 4.1.1 The architecture of an Ad hoc IR system

2 Vector Space Model

- In this the documents and queries are represented as vectors.
- Vector consists of features which represent the words present in the collection.

- Consider the example of Potato fries out recipe on internet oil in this recipe if the terms Potato, fires and salt occur term frequencies 8, 2, 7 and 4 respectively then the vector can be written as

$$\vec{d}_j = (8, 2, 7, 4)$$

- So in general for document d_j, a vector is written as,

$$\vec{d}_j = (w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{n,j})$$

where $\vec{d}_j \rightarrow$ A document

$w_{n,j} \rightarrow$ Weight that term n in document j.

- Query can be written in same way

  so query q for Potato fries can be written as,

$$\vec{q} = (1, 1, 0, 0)$$

- In general it is written as,

$$\vec{q} = (w_{1,q}, w_{2,q}, w_{3,q}, \ldots, w_{n,q})$$

  $N \rightarrow$ Dimensions in vector = Total number of terms in complete collection.
- Now consider another document recipe for Potato curry. Let's consider the vector as,

$$\vec{d}_k = (6, 0, 0, 0)$$

- If the query is written for Potato fries then it should match document d_j then to d_k.
- If we use features and queries representing a document as dimensions in multidimensional space then feature weights are used to locate documents in that particular space.
- A point in this space represents user's query translated into vector.
- This is shown graphically in Fig. 4.2.1.
- The similarity between vectors is measured by angle between the vectors.
- As shown in the Fig. 4.2.1 the query is considered more similar to d_j than to d_k as the angle between query and d_j is smaller.
- Typically a cosine metric is used in this type of retrieval.
- The distance between two documents is measured by cosine of the angle between them.

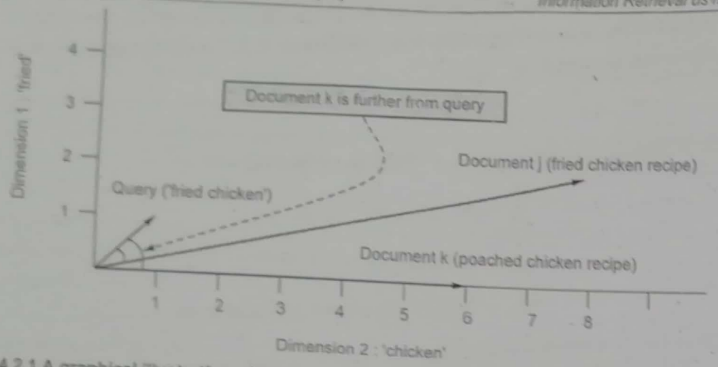


Fig. 4.2.1 A graphical illustration of the vector model for information retrieval, showing the first two dimensions (fried and chicken) and assuming that we use raw frequency in the document as the feature weights

- For identical document cosine value is 1.
- If the documents don't share common terms, the value of cosine is 0.
- The equation for cosine is written as,

$$S_{im}(\vec{q}, \vec{d_j}) = \frac{\left(\sum_{i=1}^{N} w_{i,q} \times w_{i,j}\right)}{\sqrt{\sum_{i=1}^{N} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

- Cosine can also be considered as normalized dot product i.e. it is a dot product between two vectors divided by lengths of two vectors.
- The numerator of cosine is dot product given by formula,

$$\text{Dot - product}(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_{i=1}^{N} x : x\ y :$$

- The denominator of cosine is represented by lengths of vector, where vector length is given as,

$$|\vec{x}| = \sqrt{\sum_{i=1}^{N} x_i^2}$$

- So a document retrieval system simply accepts user's query, creates vector representation, compare it with vector representing all the documents and finally the result is sorted, which is list of ranked documents according to their similarity with query.

Evaluation of IR systems :

- The basic tools for measuring performance IR system are,
  1. Precision 2. Recall.
- It is assumed that returned item is divided in two categories : The relevant item and irrelevant results.
- So precision can be stated as fraction of returned relevant documents.
- Recall is fraction of all possible relevant documents contained in return set.
- Let's consider,

  $T \rightarrow$ Total ranked documents as a result of given query

  $R \rightarrow$ Relevant documents from T

  $N \rightarrow$ Remaining irrelevant documents

  $U \rightarrow$ Relevant documents in the collection as a whole for given query.

Now the precision and recall measures can be given as,

$$\text{Friction} = \frac{|R|}{|T|}$$

$$\text{Recall} = \frac{|R|}{|U|}$$

- The drawback is, these measures are not sufficient to measure the performance of the ranked retrieval of the system as these are not dependent on rank.

Review Question

1. *Explain vector space model.*

4.3 Named Entity Recognition

- Named-Entity Recognition (NER) (also known as (named) entity identification, entity chunking, and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories
- Named entity recognition (NER) – also called entity identification or entity extraction – is a natural language processing (NLP) technique that automatically identifies named entities in a text and classifies them into predefined categories.
- Entities can be names of people, organizations, locations, medical codes, time expressions, times, quantities, monetary values, percentages and more.

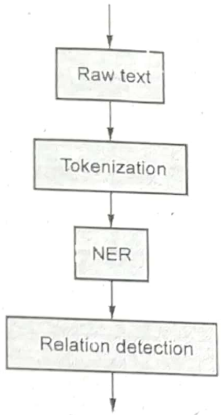


Fig. 4.3.1 Named Entity Recognition (NER) in NLP

NER example : In the sentence "Dheerubhai Ambani Founded Reliance Industries in India " we can identify three types of entities :

<Person> → Dheerubhai Ambani

<Company> → Reliance Industries

<Location> → India

- Recognition of these entities is done through machine learning and Natural Language Processing (NLP).
- With named entity recognition, one can extract key information to understand what a text is about or merely use it to collect important information to store in a database. NER systems have been created that use linguistic grammar-based techniques as well as statistical models such as machine learning. Hand-crafted grammar-based systems typically obtain better precision, but at the cost of lower recall and months of work by experienced computational linguists.
- Statistical NER systems typically require a large amount of manually annotated training data.
- Popular NER platforms include :
  1. GATE supports NER across many languages and domains out of the box, usable via a graphical interface and a Java API.
  2. OpenNLP includes rule-based and statistical named-entity recognition.
  3. SpaCy features fast statistical NER as well as an open-source named-entity visualizer.
- APIs for NER - There are TWO types of NER APIs

- Open-source named entity recognition APIs
  1. **Open-source APIs are for developers** : They are free, flexible and entail a gentle learning curve. Here are a few options :
     - **Stanford Named Entity Recognizer (SNER)** : This JAVA tool developed by Stanford University is considered the standard library for entity extraction. It's based on Conditional Random Fields (CRF) and provides pre-trained models for extracting person, organization, location and other entities.
     - **SpaCy** : A Python framework known for being fast and very easy to use. It has an excellent statistical system that you can use to build customized NER extractors.
     - **Natural Language Toolkit (NLTK)** : This suite of libraries for Python is widely used for NLP tasks. NLKT has its own classifier to recognize named entities called ne_chunk, but also provides a wrapper to use the Stanford NER tagger in python.
- SaaS named entity recognition APIs
  1. SaaS tools are ready-to-use, low-code, and cost-effective solutions. Plus, they are easy to integrate with other popular platforms.
  2. Monkey Learn, for example, is a text analysis SaaS platform that you can use for different NLP tasks, one of which is named entity recognition. You can use MonkeyLearn's ready-built API to integrate pre-trained entity extraction models or you can easily build your own custom named entity extractor in just a few simple steps.
- NER used in
  1. **Classifying content for news providers** : News and publishing houses generate large amounts of online content on a daily basis and managing them correctly is very important to get the most use of each article.

     Named entity recognition can automatically scan entire articles and reveal which are the major people, organizations and places discussed in them.

     Knowing the relevant tags for each article help in automatically categorizing the articles in defined hierarchies and enable smooth content discovery.
  2. **Efficient search algorithms** : If for every search query the algorithm ends up searching all the words in millions of articles, the process will take a lot of time. Instead, if Named Entity Recognition can be run once on all the articles and the relevant entities (tags) associated with each of those articles are stored separately, this could speed up the search process considerably.

3. **Powering content recommendations :** One of the major uses cases of named entity recognition involves automating the recommendation process.
4. **Customer support :** There are a number of ways to make the process of customer feedback handling smooth and named entity recognition could be one of them.
5. **Research papers :** An online journal or publication site holds millions of research papers and scholarly articles. There can be hundreds of papers on a single topic with slight modifications. Segregating the papers on the basis of the relevant entities it holds can save the trouble of going through the plethora of information on the subject matter.

4.3.1 NER System Building Process

- Anything which is referred by a proper name is called as **named entity.**
- Table 4.3.1 shows entity types and their categories.

Type	Tag	Sample categories
People	PER	Individuals, fictional characters, small groups.
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams.
Location	LOC	Physical extents, mountains, lakes, seas.
Geo-political entity	GPE	Countries, states, provinces, counties.
Facility	FAC	Bridges, buildings, airports.
Vehicles	VEH	Planes, trains, and automobiles.

Table 4.3.1 A list of generic named entity types with the kinds of entities they refer to

- Named Entity Recognition (NER) is a process of finding the part of text containing proper names and to identify and classify the entitie's types.
- NER systems take into account various entity types like people, places, organizations, commercial products, weapons, biological entities and many more.
- Sometimes the proper names are signaled based on their surrounding contexts.

For example :

If in a text word Dr. appears then we can say that the next word is a name of a person.

- Concept of named entity is further extended to the entities of practical importance known as temporal, for example, dates, expressions, times, named events, etc. Some represent numerical expressions like measurements, counts, prices etc.

Ambiguity in named entity recognition :

- Two types of ambiguities exist in NER.

1] The ambiguities which arise due to reference resolution problem :

- In some cases it may happen that same name can be used for different entities
- **For example :** JFK can be used to address former president or his son.

2] The ambiguities which arise due to cross type confusion :

- It may happen that same name can refer to a person as well as some school name or airport name.
- For example : In extension with previous example along with name of person JFK can also refer to a school name or airport name.

NER as sequence labeling :

- To address the problem of NER word by word sequence labeling task is used.
- In this approach the assigned tags can capture both the boundary and type of detected entities.
- The classifiers are trained for labelling the tokens in a text with tags. These tags indicate the presence of specific kinds of named entities.

Review Questions

1. *What is named entity recognition ?*
2. *What are open source named entity recognition API's ?*
3. *Explain NER system building process.*

4.4 Evaluating NER System Entity Extraction

Evaluation of named entity recognition :

- Recall, precision and F, measures matrices are used for evaluation of NER.

where

Recall ratio of the number of correctly labelled responses to total responses eligible for labelling

Precision = Ratio of the number of correctly labelled responses to total labelled

F measure = Way to combine above two measures into single matrix.

$$\text{So, } F_\beta = \frac{(\beta 2 + 1)\, PR}{\beta 2\, P + R}$$

Where $\beta \rightarrow$ Weights the importance of recall of precision.

$\beta > 1$ Favor recall

$\beta < 1$ Favor precision

β = 1 Recall and precision are balanced

In this case f can be written as $f_1 = \frac{2PR}{P+R}$

Review Question

1. *How to evaluate NLP system ?*

4.5 Reference Resolution

- To understand the concept of reference resolution, lets consider following example.
- "Lakshmi is studying in 10th standard. She is a very good singer and participated in many music programs. The performance of this 16 years old is also excellent in academics."
- In the above passage there is mention of one person named Lakshmi.
- The linguistic expressions like her, she are used to denote an individual is known as reference.
- Reference resolution is a task to determine what entities are referred to by which linguistic expressions.
- Referring expression (for ex. she) is a haliral language expression used to perform reference.
- The referred entity is called as reference (for ex. Lakshmi).
- Reference to an entity which is previously introduced in discourse is called anaphora and the referring expression is called as amaphoric. For ex. Pronoun, she and 16 years old are anaphoric.
- Two referring expressions used to refere same entity are said to corefer.
- So typically the task of reference resolution involve two tasks :
  1. Coreference resolution
  2. Pronominal anaphora resolution
- Coreference resolution is the task to find out referring expressions referring to same entity.
- Pronominal anaphora resolution is the task to find out antecedent for single pronoun.
- For ex. : antecedent of she is Lakshmi. We need to find out the given a pronoun she, its antecedent is Lakshmi.

- Pronominal anaphora resolution can be considered as subtask of coreference resolution.
- There are various algorithms which can be used for this purpose. Some of them are :
  1. Hobbs algorithm
  2. Centering algorithm
  3. Log linear algorithm

Review Question

1. *What is reference resolution ?*

4.6 Coreference Resolution

- Coreference resolution is the task of finding all expressions that refer to the same entity in a text. It is an important step for a lot of higher level NLP tasks that involve natural language understanding such as document summarization, question answering and information extraction.
- Coreference Resolution in particular, is the process of resolving pronouns to identify which entities are they referring to. It is also a kind of Reference Resolution. The entities resolved may be a person, place, organization, or event.

Review Question

1. *Explain coreference resolution.*

4.7 Cross Lingual Information Retrieval (CLIR)

- Humans have the unique ability to express themselves through different mediums of communication like : Oral, written, through images and pictures, through gestures and sign language, etc.
- Out of these, oral and written communication in natural language facilitates the fastest way of conveying the ideas, so they are very popular.
- Written communication is advantageous as it preserves the information manually as well as electronically.
- Around 7000 languages exist in the world, out of which very few are used across the globe.

- English language has the greatest internet presence, even though it is not the most spoken language in the world.
- With the advancements in technology, a humongous amount of e-text is getting generated across the world.
- The e-text may contain :
  1. Documents in different languages
  2. Multilingual documents
  3. Images with captions in different languages.
- People with linguistic diversity have started using their mother tongue for searching the documents present in various domains and languages.
- While using search engines, query based information retrieval is a common way.
- With this context in simple words Cross-Lingual Information Retrieval (CLIR) is retrieving information in one language based on the query written in another language.
- The users can search document databases in multiple languages and retrieve information in a form that is useful to them, even though they don't have linguistic competence in the target languages.
- CLIR is important for countries like India where a very large fraction of people are not conversant with English and thus don't have access to the vast store of information on the web.
- Searching distributed, unstructured, heterogeneous, multilingual data is the goal of CLIR system.

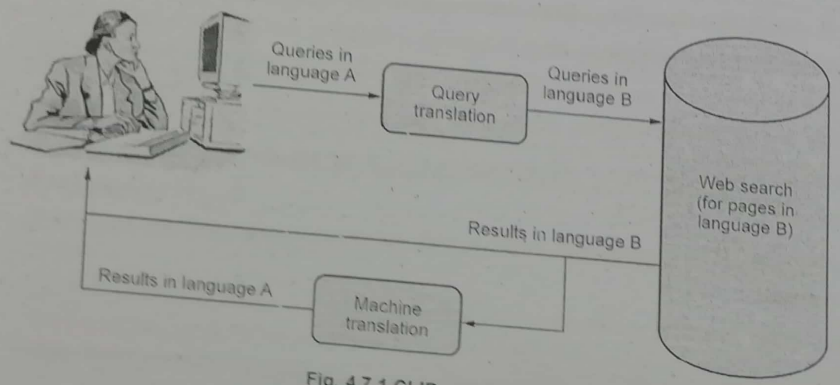


Fig. 4.7.1 CLIR system

- As shown in the Fig. 4.7.1, a user can enter a query in any language. For example Hindi. If the relevant information is present in language B (For example English) the query is first translated to language B. After the web search the retrieved information in language B is again translated to language A, and results are displayed to the user.

Different approaches of CLIR are :

1. Query translation approach :

- As shown in the Fig. 4.7.2, the query is translated to the target document language.
- This is the most appropriate approach, as the query is shorter and fast to translate than a complete document.
- One disadvantage can be, query translation can suffer from translation ambiguity due to the limited context.

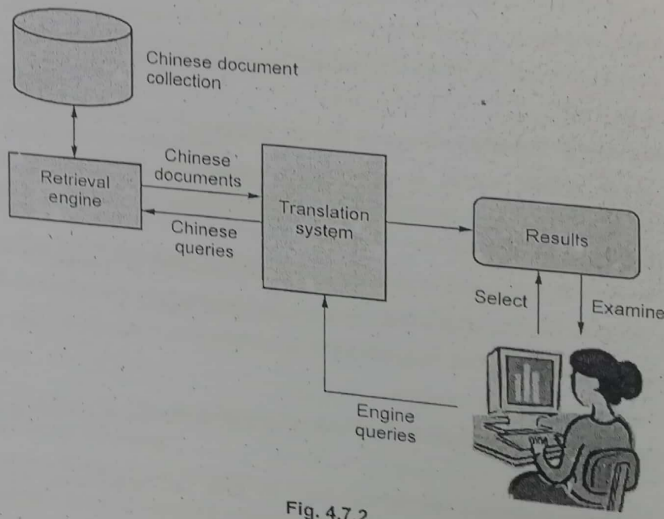


Fig. 4.7.2

2. **Document translation approach :**

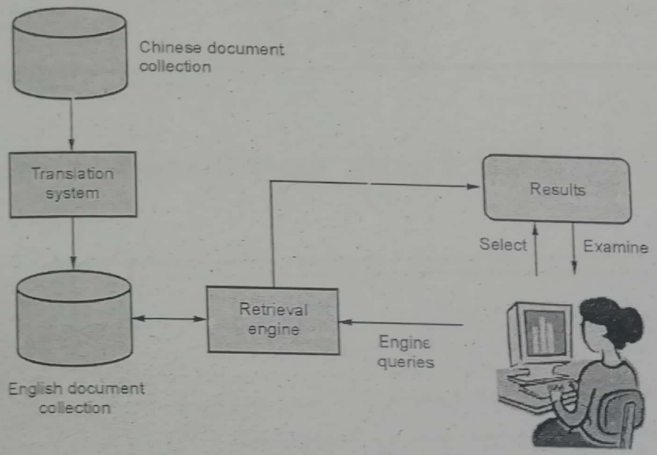


Fig. 4.7.3

- As shown in the Fig. 4.7.3, the documents in target language are translated to source language.
- Document translation can provide more accurate translation due to richer contexts.
- Advantage of document translation is, it's easy for the user to understand the document easily once it is retrieved.

3. **Interlingua based approach :**

- In this the documents and the query are both translated into some common third Interlingua (like UNL).
- This approach generally requires huge resources as the translation needs to be done online.

Challenges in CLIR :

1. **Translation ambiguity :**

- Due to ambiguities in various phases of NLP, more than one translation is possible of the source sentence.
- **For example :** Word "Pooja" can have two meanings, it can be the name of a girl or the second meaning can be worshipping.

2. **Phrase identification :**

- In some languages like Japanese the words are not separated by white spaces. This may lead to the incorrect translation.

3. **Dictionary coverage :**

- The linguistic resources in the dictionary can prove as a constraint in performance of the system.
- Out-of-Vocabulary (OOV) problems.
- Daily new words are added to the language, which may not be recognized by the system.
- Apart from traditional CLIR systems, recently cross-lingual word embeddings and neural network based information retrieval systems are becoming increasingly popular.
- Cross-lingual word embeddings can represent words in different languages in the same vector space by learning a mapping from monolingual embeddings.
- Neural network based information retrieval can produce better representations for documents and queries. In this case the ranking is done directly from relevant labels.

Review Question

1. *Explain cross lingual information retrieval.*

❑❑❑

Notes

Unit V

5 NLP Tools and Techniques

Syllabus

Prominent NLP Libraries : *Natural Language Tool Kit (NLTK), spaCy, TextBlob, Gensim etc.* ***Linguistic Resources :*** *Lexical Knowledge Networks, WordNets, Indian Language WordNet (IndoWordnet), VerbNets, PropBank, Treebanks, Universal Dependency Treebanks.* ***Word Sense Disambiguation :*** *Lesk Algorithm Walker's algorithm, WordNets for Word Sense Disambiguation.*

Contents

5.1 Prom

Here i
applicatio
However
libraries.

Core NLI

- St
  It
  e
  e
-

Scil

Py

N

5.1 Prominent NLP Libraries : NLTK

Here is a list of NLP libraries that can be readily used while coding an NLP application. There are numerous NLP libraries designed for specific NLP applications. However Python is the lead programming language that supports most of the NLP libraries. Here is a list of few popular NLP libraries.

Core NLP :

- Stanford CoreNLP comprises of an assortment of human language technology tools. It aims to make the application of linguistic analysis tools to a piece of text easy and efficient. With CoreNLP, you can extract all kinds of text properties (like named-entity recognition, part - of - speech tagging, etc.) in only a few lines of code.
- The tool incorporates numerous Stanford's NLP tools like the parser, sentiment analysis, bootstrapped pattern learning, Part - Of - Speech (POS) tagger, Named Entity Recognizer (NER) and coreference resolution system, to name a few. Furthermore, CoreNLP supports four languages apart from English - Arabic, Chinese, German, French and Spanish.

Scikit - learn :

It provides a wide range of algorithms for building machine learning models in ython.

atural Language Tool Kit (NLTK) :

NLTK is a complete toolkit for all NLP techniques. NLTK comes with a host of text ocessing libraries for sentence detection, tokenization, lemmatization, stemming, rsing, chunking, and POS tagging.

tern :

Pattern is a text processing, web mining, natural language processing, machine ning, and network analysis tool for Python. It comes with a host of tools for data ing (Google, Twitter, Wikipedia API, a web crawler and an HTML DOM parser), NLP -of-speech taggers, n-gram search, sentiment analysis, WordNet), ML (vector space el, clustering, SVM) and network analysis by graph centrality and visualization.

ew Question

Explain NLTK.

5.2 SpaCy

- SpaCy is an open - source NLP library which is used for data extraction, data analysis, sentiment analysis, and text summarization. SpaCy is an open-source NLP library in Python. It is designed explicitly for production usage - it lets you develop applications that process and understand huge volumes of text.
- SpaCy can preprocess text for Deep Learning. It can be used to build natural language understanding systems or information extraction systems. SpaCy is equipped with pre-trained statistical models and word vectors. It can support tokenization for over 49 languages. SpaCy boasts of state - of - the - art speed, parsing, named entity recognition, convolutional neural network models for tagging and deep learning integration.

Review Question

1. *What is SpaCy.*

5.3 TextBlob

- It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos - tagging. TextBlob is a Python (2 and 3) library designed for processing textual data. It focuses on providing access to common text - processing operations through familiar interfaces. TextBlob objects can be treated as Python strings that are trained in Natural Language Processing.
- TextBlob offers a neat API for performing common NLP tasks like part - of - speech tagging, noun phrase extraction, sentiment analysis, classification, language translation, word inflection, parsing, n - grams, and WordNet integration.

PyNLPl :

- Pronounced as 'pineapple,' PyNLPl is a Python library for Natural Language Processing. It contains a collection of custom-made Python modules for Natural Language Processing tasks. One of the most notable features of PyNLPl is that it features an extensive library for working with FoLiA XML (Format for Linguistic Annotation).
- PyNLPl is segregated into different modules and packages, each useful for both standard and advanced NLP tasks. While you can use PyNLPl for basic NLP tasks like extraction of n-grams and frequency lists and to build a simple language model, it also has more complex data types and algorithms for advanced NLP tasks.

Quepy :

Quepy is used to transform natural language questions into queries in a database query language.

Review Questions

1. *Explain TextBlob.*
2. *Explain PyNLPI.*

5.4 Gensim

- Gensim works with large datasets and processes data streams. Gensim is a Python library designed specifically for "topic modeling, document indexing and similarity retrieval with large corpora." All algorithms in Gensim are memory-independent, w.r.t., the corpus size and hence, it can process input larger than RAM.
- With intuitive interfaces, Gensim allows for efficient multicore implementations of popular algorithms, including online Latent Semantic Analysis (LSA / LSI / SVD), Latent Dirichlet Allocation (LDA), Random Projections (RP), Hierarchical Dirichlet Process (HDP) or word2vec deep learning.

Review Question

1. *What is Gensim ?*

5.5 Linguistic Resources : Lexical Knowledge Networks

- Lexical knowledge is essential for semantic analysis .
- Traditionally, semantic information in computational lexicons is limited to notions such as selectional restrictions or domain-specific constraints, encoded in a "static" representation.
- Lexical Knowledge Networks (LKN) are resources which are critical for most NLP applications.
- This is evidenced by the success of the efforts on wordnets which are now regarded as indispensable for work on semantic search, knowledge based machine translation, summarization, question answering and such other high end NLP applications.
- Lexical Knowledge Bases (LKBs), also known as **lexico-semantic resources**, provide information about words and potentially entities and are at the core of knowledge-based approaches.
- They are widely used in a variety of NLP tasks (e.g., word sense disambiguation, information retrieval and question answering), all the more so since their traditional

limitations (i.e., lack of language and domain-specific coverage) have started to fall.

- Beside the long-established process of expert-based resource creation (e.g., WordNet), web technologies have enabled the collaborative, crowd-based construction of resources (e.g., Wikipedia and Wiktionary).
- This contributed to significantly widen the scope of the available machine-readable knowledge and in the context of an already diverse landscape of LKBs, encouraged and motivated even more the need to integrate different resources so as to make the best of them all.
- Parameters vital for building Lexical Knowledge Networks are,
  - Domains addressed by the structure
  - Principles of construction
  - Methods of construction
  - Representation
  - Quality of database
  - Applications
  - Usability mechanisms for software applications and users : APIs, record structure, User interfaces
  - Size and coverage.

IndoNet as an example of Lexical knowledge networks

- IndoNet is a multilingual lexical knowledge base for Indian languages.

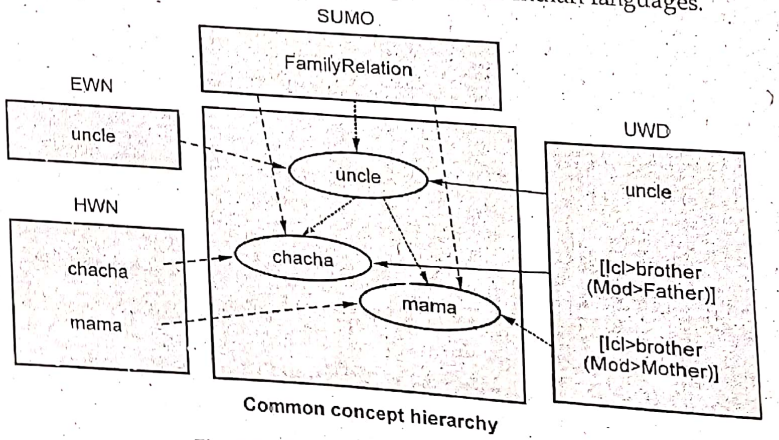


Fig. 5.5.1 Example of IndoNet structure

- It is a linked structure of wordnets of 18 different Indian languages, Universal Word dictionary and the Suggested Upper Merged Ontology (SUMO).
- IndoNet is encoded in Lexical Markup Framework (LMF), an ISO standard (ISO-24613) for encoding lexical resources.

Review Questions

1. Explain Lexical knowledge networks.

5.6 WordNets

- In this world people communicate with each other in more than 7000 languages.
- Written communication is the most popular form of communication.
- Any text comprises sentences which are formed by basic text units i.e words.
- Automatic understanding, analysis and preprocessing of text and in turn words is a challenging task for the NLP system.
- Typically this is done with the help of lexicons.
- A lexicon is a vocabulary of a person, language or branch of knowledge.
- Text in the textual data is mapped to the lexicon which helps us to understand the relationships between the words.
- WordNet is the lexical resource which helps us find word relations, synonyms, grammers, etc.
- WordNet is useful in solving many NLP applications like machine translation, sentiment analysis, etc.
- WordNet is a network of words linked by lexical and semantic relations and is freely and publicly available.
- The words in WordNet are related by synonymy, for example the words shut and close or car and automobile possess the same concept and can be interchanged as per the context. All such words are grouped into unordered sets or synsets. Under synsets nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms.

Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet can be confused with thesaurus as, just like thesaurus it is collection words based on their meaning but following are the unique points of WordNet :

- Along with word forms WordNet also interlinks specific senses of words. Due to this, words that are found in close proximity to one another in the network are semantically disambiguated.

- WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.

- The major relations which are covered in WordNet are shown in below Table 5.6.1 :

Semantic Relation	Syntactic Category	Examples
Synonymy (similar)	N, V, Aj, Av	Pipe, tube rise, ascend sad, unhappy rapidly, speedily.
Antonymy (opposite)	Aj, Av, (N, V)	Wet, dry powerful, powerless friendly, unfriendly rapidly, slowly.
Hyponumy (subordinate)	N	Sugar maple, maple maple, tree tree, plant.
Meronymy (part)	N	brim, hat gin, martini ship, fleet.
Troponomy (manner)	V	March, walk whisper, speak
Entailment	V	Drive, ride divorce, marry
Note : N = Nouns Aj = Adjectives V = Verbs Av = Adverbs		

Fig. 5.6.1

- Reference : Semantic relations in WordNet. Source : Miller 1995, Table 5.6.1.
- Following are meanings of the relations mentioned in the table :
  - Synonymy : Words with the same meaning.
  - Polysemy : Words with more than one sense.
  - Hyponymy/Hypernymy : Words which are having is-a relation.
  - For example : Peacock is a type of bird. Likewise, bird is a hypernym of Peacock.
  - Meronymy/Holonymy : Words with part-whole relation. For example : Beak is meronym of bird since beak is part of a bird's anatomy. Likewise, bird is holonym of beak.
  - Antonymy : Words which are lexically opposite. For example : Hot, cold.
  - Troponymy : Applies to verbs. For example, whisper is a troponym of talk since whisper elaborates on the manner of talking.

- Many WordNets are built across the world for various languages. Few of them are listed in the below table :

Language	Resource name	Developer(s)
Multilingual (South African languages/isiZulu, isiXhosa, Setswana, Sesotho sa Leboa, Tshivenda, Sesotho, isiNdebele, Siswati & Xitsonga	African Wordnet (AfWN)	University of South Africa & South African Centre for Digital Language Resources, Pretoria, South Africa
Multilingual (Hindi/ Indonesian/ Japanese/ Lao/ Mongolian/ Burmese/ Nepali/ Sinhala/ Thai/ Vietnamese)	Asian WordNet	National Electronics and Computer Technology Center (NECTEC) – Thai Computational Linguistics Laboratory (TCL) – NICT, Kyoto, Japan
Multilingual (English/ Spanish/ Catalan/ Basque/ Italian)	Multilingual Central Repository	University of the Basque Country – Department of Software, Technical University of Catalonia (UPC)
English	WordNet 3.01	Princeton University
Hindi, Marathi, Sanskrit	Hindi WordNet	Indian Institute of Technology Bombay Powai, Mumbai
	Marathi WordNet	
	Sanskrit Wordnet	

5.7 Indian Language WordNet

- IndoWordNet (Bhattacharyya, P. (2010). Indowordnet. In Proc. of LREC-10. Citeseer.) is a WordNet consisting of wordnets for major Indian languages.
- The languages covered are : Assamese, Bengali, Bodo, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Malayalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Tamil, Telugu, and Urdu.
- These wordnets have been created by expanding the expansion approach of Hindi WordNet as a pivot, which is partially linked to English WordNet.

- The expansion approach adopted for IndoWordNet creation is as follows (https://aclanthology.org/L18-1728.pdf) :
  1. Creation of a Hindi synset with synonymous words.
  2. Mapping of the synset with relations such as hypernymy and hyponymy etc.
  3. Tagging of the synset with an ontological category.
  4. Allotment of a unique synset ID to the concept de4604 scribed in the synset.
  5. Creation of the same synset in the other Indian languages leading to an implicit linkage of relations, ontological categories.

Review Question

1. *Explain Indian Language WordNet.*

5.8 VerbNet

- VebNet (VN), developed by kipper and Schuler is the largest on-line verb lexicon currently available for English. (Reference : Schuler, Karin Kipper, "VerbNet : A broad-coverage, comprehensive verb lexicon" (2005))
- VN is a domain independent broad coverage verb lexicon which has mapping to other lexical resources like WordNet, FrameNet, etc.
- It is a verb lexicon compatible with WordNet but with explicitly stated syntactic and semantic information.
- It uses Levin verb classes to systematically construct lexical entries.
- VN is organized into verb classes to achieve syntactic and semantic coherence among members of a class.
- Each verb class in VN is completely described by thematic roles, selectional restrictions on the arguments and frames consisting of a syntactic description and semantic predicates with a temporal function.
- Classes are hierarchically organized to ensure that all their members have common semantic and syntactic properties.

Review Question

1. *Explain VerbNet.*

5.9 PropBank

- PropBank is a corpus that is annotated with verbal propositions and their arguments - a "proposition bank".
- Although "ProbBank" refers to a specific corpus produced by Martha Palmer et al., the term PropBank is also coming to be used as a common noun referring to any corpus that has been annotated with propositions and their arguments.
- PropBank is a corpus of text annotated with information about basic semantic propositions created by Martha Palmer.
- It is created by adding predicate-argument relations to the syntactic trees of Penn Treebank.
- In PropBank the arguments of each predicate are annotated with their semantic roles in relation to the predicate.
- PropBank annotation also requires the choice of a sense id (also known as a 'frameset' or 'roleset' id)for each predicate.
- For each verb in every tree which presents the phrase structure of the sentence, a PropBank instance is created which consists of the sense id of the predicate and its arguments labeled with semantic roles.
- It provides consistent argument labels across different syntactic realizations of the same verb.

For e.g. : [ARG0 Devika]broke [ARG1 the window]

[ARG1 The window] broke

In the above example the arguments of the verbs are labeled as numbered arguments : Arg0, Arg1, Arg2 and so on.

As shown in Table 5.9.1 the arguments are tagged as core arguments (label type ARGN, where N = 0 to 5) or adjunctive arguments (label type ARGM-X, where X = TMP for temporal, LOC for locative, etc.)

Table 5.9.1 : List of adjunctive arguments in PropBank - ARGMS

Tag	Description	Example
ARGM-LOC	Locative	*the museum, in Westborough, Mass*
ARGM-TMP	Temporal	*now, by next summer*
ARGM-MNR	Manner	*heavily, clearly, at a rapid rate*
ARGM-DIR	Direction	*to market, to Bangkok*
ARGM-CAU	Cause	*In response to the ruling*
ARGM-DIS	Discourse	*for example, in part, Similarly*
ARGM-EXT	Extent	*at $38.375, 50 points*
ARGM-PRP	Purpose	*to pay for the plant*
ARGM-NEG	Negation	*not, n't*
ARGM-MOD	Modal	*can, might, should, will*
ARGM-REC	Reciprocals	*each other*
ARGM-PRD	Secondary predication	*to become a teacher*
ARGM	Bare ARGM	*with a police escort*
ARGM-ADV	Adverbials	*(none of the above)*

- For e.g. Core arguments for predicates operate and author are shown in Table 5.9.2.

Table 5.9.2 : Argument labels associated with the predicate operate.01 (sense : work) and for author.01 (sense : to write or construct) in the PropBank corpus

Predicate	Argument	Description
operate.01		
	ARG0	Agent, operator
	ARG1	Thing operated
	ARG2	Explicit patient (thing operated on)
	ARG3	Explicit argument
	ARG4	Explicit instrument
author.01		
	ARG0	Author, agent
	ARG1	Text authored

Review Question

1. *What is PropBank ?*

5.10 Treebanks

- Due to ambiguous nature of a language two knowledge acquisition problems arise in syntaclic analysis.
  1. Due to difficulty in exploration of all the possibilities based on part of speech tags mentioned for a particular word.
  2. Due to difficulty in understanding the feasible meaning of sentence due to ambiguity in syntactic analysis.
- To address these problems a data driven approach called as **treebank** can be adapted.
- Treebank is collection of sentences.
- Each sentence in treebank contains complete syntax analysis.
- Human expert provides feasible analysis of the sentence. Annotation guidelines are provided before annotation process.
- Treebank contains annotations of syntactic structure for large set of sentences.
- Superviced machine learning techniques are used to train the parser from the training data extracted from tree banks.
- The first knowledge acquition proolem is addressed by providing syntactic analysis directly instead of grammer.
- The second knowledge acquisition problem is solved as for each sentence in a treebank the most feasible syntactic analysis is provided.
- Using supervised machine learning techniques are used to learn scoring function for all possible syntax analysis.

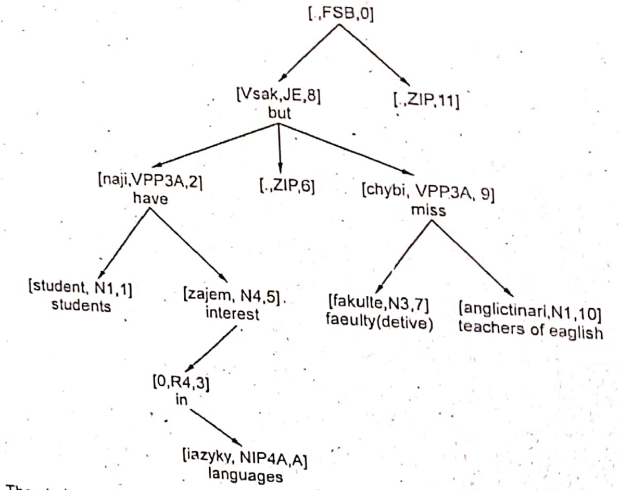
Review Question

1. *Explain Treebanks.*

5.11 Universal Dependency Treebanks

- Dependency graphs connects head of a phrase to its dependents.
- According to definition of Co NLL, in dependency graph nodes are words of input sentence and arcs are binary relations from head to dependent.

- It labelled dependency parsing a label is assigned to each dependency relation between head and dependent word.
- The example dependency graph for Czech sentence from Prague Dependency Treebank is shown in Fig. 5.11.1.
- It is observed that dependency analysis make minimal assumption about syntactic structure for avoiding annotation of hidden structure like empty elements to represent missing arguments of predicates.



The students are interested In languages, but the faculty is missing teachers of English

Fig. 5.11.1 : An example of a dependency graph syntax analysis for a Czech sentence taken from Prague Dependency Treebank. Each node in the graph is a word, its part of speech, and the not of the word in the sentence, for example [fakulte, N3, 7] is the seventh wood In the sentence with F tag N3, which also tells us that the word has dative case. The node [#, ZSB,0] is the root node of dependency tree. The English equivalent is provided for each node

- Projectivity is the constraint on syntactic analysis due to effect of linear order of words on dependencies between words.
- The example shown in Fig. 5.11.2 shows the english sentence with the requirements of crossing dependencies.

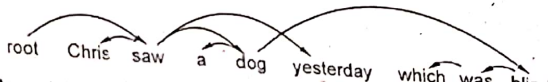


Fig. 5.11.2 : An unlabeled nonprojective dependency tree with a crossing dependency

5.12 WSD : I

- In early
  definiti
- These a
  availat
- Despit
  techn
- The f
  algor
- Lesk

5.12 WSD : Lesk Algorithm

- In early days word sense disambiguation used to done using dictionary sense definitions.
- These are the man made resources and now this has become historical data which is available in archived publication.
- Despite of the fact that this information is not available today, some algorithms and techniques are still useful.
- The first and very simple algorithm was proposed by lesk and these first generation algorithms are based on computerized dictionaries.
- Lesk algorithm is further simplified as shown in Fig. 5.12.1 Algorithm 5.12.1.

Algorithm 5.12.1 : Pseudocode of the simplified Lesk algorithm
The function COMPUTEOVERLAP returns the number of words common to the two sets
Procedure : SIMPLIFIED_LESK (word, sentence) **returns** best sense of word

```
best-sense ← most frequent sense of word
max-overlap ← 0
context ← set of words in sentence
for all sense ∈ senses of word do
    signature ← set of words in gloss and examples of sense
    overlap ← COMPUTEOVERLAP (signature, context)
    if overlap gt max-overlap then
        max-overlap ← overlap
        best-sense ← sense
    end if
end for
return best-sense
```

Fig. 5.12.1 : Algorithm 5.12.1

Review Question

1. *Explain Lesk Algorithm.*

5.13 Walker's Algorithm

- To understand the concept of word sense disambiguation let's consider following example.

1. I went to bank to withdraw money.
2. I went to bank to fetch water.

- In the above example the same word 'bank' has appeared in two different senses and contexts.
- In computational lexical semantics it is required to examine the contextual word tokens and to figure out which sense of word is used.
- This task is known as Word Sense Disambiguation (WSD).
- WSD is the task of selecting the correct sense for a word.
- WSD is essential part of many important NLP applications like question answering, information retrieval and text classification where the use of wrong senses of words can create disasters.
- Typically any WSD algorithms, input is a word in context and fixed inventory of possible word senses. The output is a correct word sense.
- The task for which WSD is done decides nature of input and inventory of senses used.
- For example : If the task is of machine translation from English to Spanish, sense tag inventory for an English word can be set of Spanish translations. But if the task is automatic indexing of medical articles inventory can be MeSH (Medical Subject Headings) thesaurus entries.

Review Question

1. *Explain Walker's algorithm.*

5.14 WordNets for WSD

- The main requirement of supervised WSD is we need labeled data.
- If we have hand labeled data with proper word senses, then using supervised WSD can be used to extract features from the text.
- These features can be used to predict the senses and a classifier can be learned for assigning correct senses from the features.
- These are two broad steps in supervised WSD :
  1. Creative extraction for supervised learning.
  2. Training a classifier.

1. Feature extraction for supervised learning :

- In this step features, predicting word sense are extracted.
- First preprocessing is performed which includes POS tagging, stemming and lemmatization.

So, $$P(s_i) = \frac{\text{count}(s_i, w_j)}{\text{count}(w_j)}$$

- So in our example of 'bass' if collocational feature guitar occurs 9 times for sense 'bass' and if sense "bass" occurs 60 times in training the maximum likelihood estimate is $P(l_j|s) = 0.05$

2) Decision list classifier :

- In this, a sequence of tests is applied to each target word feature vector
- A specific sense is shown by each test
- A sense is returned with a successful test and in case of failure the next test is applied till end of list
- Consider the disambiguation of the sense 'bass' from Fig. 5.14.1
- As shown in Fig. 5.14.1 the first test indicates that if word fish occurs in input context then bass in correct

Rule		Sense
fish within window	⇒	$bass^1$
stripped bass	⇒	$bass^1$
guitar within window	⇒	$bass^2$
bass player	⇒	$bass^2$
piano within window	⇒	$bass^2$
tenor within window	⇒	$bass^2$
sea bass	⇒	$bass^1$
play/V bass	⇒	$bass^2$
river within window	⇒	$bass^1$
violin within window	⇒	$bass^2$
salmon within window	⇒	$bass^1$
on bass	⇒	$bass^2$
bass are	⇒	$bass^1$

An abbreviated decision list for disambiguating the fish sense of bass from the music sense (Yarowsky, 1997)

se fish doesn't occur next text is carried out till we obtain the result as true

efault test returning trade is mentioned at the last in the list