USN | | | | | | | | | |

Internal Assessment Test 3 – JULY 2024

Sub:	Natural Language Processing				Sub Code:	21AI643	Branch:	AIML
Date:	30/07/2024	Duration:	90 mins	Max Marks: 50	Sem / Sec:	VI/ A		OBE

Answer any FIVE FULL Questions — MARKS | CO | RBT

1 Describe the following:

a) Topic models

b) Cohesion and Cohesion Matrix

[5+5] Des-5 eg-2 | CO4 | L2

Topic modeling is a technique in natural language processing (NLP) used to identify the underlying topics within a collection of documents. It helps in understanding the structure and themes present in large text datasets. Here are a few examples of topic modeling algorithms:

1. **Latent Dirichlet Allocation (LDA):**
   - LDA is a generative probabilistic model that assumes documents are mixtures of topics, and topics are mixtures of words. It helps in discovering hidden topics in a set of documents.
   - **Example:** Given a set of news articles, LDA might identify topics such as politics, sports, technology, and health, with each topic represented by a set of words like:
     - Politics: election, government, policy, vote
     - Sports: game, team, player, match
     - Technology: software, computer, internet, device
     - Health: doctor, disease, treatment, medicine
2. **Non-Negative Matrix Factorization (NMF):**
   - NMF factorizes the term-document matrix into two non-negative matrices: one representing the document-topic distribution and the other representing the topic-word distribution.
   - **Example:** In a collection of research papers, NMF might identify topics like machine learning, data science, and biology, with each topic characterized by specific terms:
     - Machine Learning: algorithm, model, training, neural
     - Data Science: data, analysis, statistical, visualization
     - Biology: cell, DNA, protein, organism
3. **Latent Semantic Analysis (LSA):**
   - LSA uses singular value decomposition (SVD) to decompose the term-document matrix and capture the latent relationships between terms and documents.
   - **Example:** Analyzing customer reviews, LSA might reveal topics such as product quality, customer service, and delivery experience, with each topic identified by relevant terms:
     - Product Quality: durable, reliable, excellent, value
     - Customer Service: helpful, support, response, friendly
     - Delivery Experience: fast, timely, package, tracking

Cohesion

dec-3 eg-2

Cohesion refers to the degree to which elements within a text or set of texts are semantically related or stick together in a meaningful way. In topic modeling, cohesion usually pertains to the semantic similarity of words within a topic. Higher cohesion indicates that the words within a topic are closely related in meaning, which makes the topics more interpretable.

Example:

- A topic with high cohesion might include words like "doctor," "nurse," "hospital," and "patient," all related to the theme of healthcare.
- A topic with low cohesion might include unrelated words like "doctor," "car," "software," and "economy," making it harder to interpret the theme.

Cohesion Matrix

A **cohesion matrix** (or coherence matrix) is used to evaluate the quality of the topics generated by a topic model. It provides a numerical representation of the cohesion between different elements (e.g., words within a topic).

Types of Cohesion Measures:

1. **Pointwise Mutual Information (PMI):**

- PMI measures the association between pairs of words within a topic. Higher PMI values indicate that words co-occur more frequently than would be expected by chance.
- **Example:** If the words "doctor" and "patient" frequently co-occur in documents about healthcare, their PMI would be high, indicating strong cohesion.

2. **Cosine Similarity:**
   - Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, indicating how similar the vectors are.
   - **Example:** In a vector space model, if the word vectors for "computer" and "software" are close to each other, their cosine similarity would be high, suggesting they belong to a cohesive topic.
3. **Word Co-occurrence:**
   - This measures how often words appear together within a certain context window in the text. High co-occurrence rates indicate strong cohesion.
   - **Example:** In a collection of articles about technology, the words "internet" and "browser" might frequently appear together, indicating a cohesive topic.

Applications of Cohesion and Cohesion Matrix:

- **Model Evaluation:** Assessing the quality and interpretability of the topics generated by a topic model.
- **Parameter Tuning:** Helping in the selection of optimal parameters for topic modeling algorithms, such as the number of topics.
- **Comparison of Models:** Comparing different topic modeling approaches or algorithms to determine which provides more coherent and meaningful topics.

- Texts contain pertinent information that co-refers across sentences and paragraphs,texts contain relations between phrases, clauses, and sentences that are often causally linked; and texts that depend on relating a series of chronological events contain temporal features that help the reader to build a coherent representation of the text.

For example in the movie “The Avengers”, when the helicarrier is attacked and one of its engine is broken, and iron man and captain america are trying to fix it, the conversation between them goes like this:

That stator control unit can reverse the polarity long enough to disengage maglev and that could..

Speak English!

You see that red lever? It will slow the rotors down long enough for me to get out. Stand by it. Wait for my word.

- But cohesive elements, and by consequence cohesion, does not simply feature in a text as dialogues tend to feature in narratives, or as cartoons tend to feature in newspapers.
- That is, cohesion is not present or absent in a binary or optional sense. Instead, cohesion in text exists on a continuum of presence, which is sometimes indicative of the text-type in question and sometimes indicative of the audience for which the text was written.
- In this paper, we discuss the nature and importance of cohesion; we demonstrate a computational tool that measures cohesion; and, most importantly, we demonstrate a novel approach to identifying text-types by incorporating contrasting rates of cohesion.

2	Explain Literal word matching and Soundex word matching approaches. Apply the word matching algorithms with the given string below: word1 = "example" word2 = "exampel"	[10] dec-5 sol-5	CO4	L3

Literal Word Matching

Literal word matching refers to comparing two words character by character to check if they are exactly the same. This is a straightforward method and does not account for minor spelling errors or variations.

Example:

- Given words: `word1 = "example"` and `word2 = "exampel"`
- Literal matching will compare each character in both words.
  - e vs e
  - x vs x
  - a vs a
  - m vs m
  - p vs p
  - l vs e (mismatch)
  - e vs l (mismatch)

Since there are mismatches, `word1` and `word2` do not match literally.

Soundex Word Matching

Soundex is a phonetic algorithm used to index words by their sound when pronounced in English. It maps words to a four-character code based on their pronunciation. Words that sound similar should have the same Soundex code, even if they are spelled differently.

Soundex Algorithm Steps:

1. Retain the first letter of the word.
2. Replace the remaining letters with numbers based on the Soundex table.
3. Remove all occurrences of a, e, i, o, u, y, h, w (except the first letter).
4. Replace consecutive duplicate numbers with a single number.
5. Truncate or pad the result to ensure it is four characters long.

Example:

- Given words: `word1 = "example"` and `word2 = "exampel"`
- Convert each word to its Soundex code:
  For `word1 = "example"`:
  - E (retain first letter)
  - x = 2
  - a (ignore)
  - m = 5
  - p = 1
  - l = 4
  - e (ignore)
  - Resulting code: E251
- For `word2 = "exampel"`:
  - E (retain first letter)
  - x = 2
  - a (ignore)
  - m = 5
  - p = 1
  - e (ignore)
  - l = 4
  - Resulting code: E251

Both words have the same Soundex code `E251`, indicating they sound similar.

Applying the Algorithms to the Given Strings

Literal Word Matching:

- `word1 = "example"`
- `word2 = "exampel"`
- Literal match result: **No match** (since there are mismatches at the 6th and 7th positions).

Soundex Word Matching:

- `word1 = "example"`
- `word2 = "exampel"`
- Soundex codes: Both are `E251`.
- Soundex match result: **Match** (since both words have the same Soundex code).

In summary, while literal word matching identifies `word1` and `word2` as different, Soundex word matching identifies them as similar based on their phonetic representation.

Q	Question	Marks	CO	Level
3	Describe the following approaches used in Information Retrieval with suitable examples and algorithms. a) Indexing b) Stop words elimination c) Stemming	[10] des-7 example-3	CO5	L2

1. Indexing

Indexing is the process of creating data structures that allow for efficient retrieval of information from a large dataset. In the context of information retrieval, indexing involves organizing documents or data in a way that makes it easier and faster to search and retrieve relevant information.

Types of Indexing:

- **Inverted Index:** An inverted index maps terms to the documents that contain them. This allows for quick lookup of documents based on the presence of specific terms.
  Algorithm:
  - **Tokenization:** Break down documents into individual terms or tokens.
  - **Normalization:** Convert tokens to a standard form (e.g., lowercase).
  - **Index Creation:** Create a dictionary where each term points to a list of documents (and optionally, positions within the documents) that contain the term.
- **Example:** Consider the following documents:
  - Document 1: "Information retrieval is fun"
  - Document 2: "Retrieval of information is important"

The inverted index would be:
json

```
{
  "information": [1, 2],
  "retrieval": [1, 2],
  "is": [1, 2],
  "fun": [1],
  "of": [2],
  "important": [2]
}
```

2. Stop Words Elimination

Stop words elimination is the process of removing common words that are unlikely to be useful in retrieving relevant documents. These words (e.g., "the," "is," "in," "and") occur frequently but do not carry significant meaning in the context of a search query.

Algorithm:

1. **Create a List of Stop Words:** This list contains words to be removed during preprocessing.
2. **Tokenization:** Break down documents into individual terms.
3. **Stop Words Removal:** Remove terms that are present in the stop words list.

Example: Given the document: "Information retrieval is fun and important"

- Stop words list: ["is", "and"]
- After removal: "Information retrieval fun important"

3. Stemming

Stemming is the process of reducing words to their base or root form. The goal is to treat words with the same root as identical for retrieval purposes, which helps in matching documents to queries more effectively.

Types of Stemming Algorithms:

- **Porter Stemmer:** One of the most widely used stemming algorithms, which applies a series of rules to iteratively reduce words to their stems.
  Algorithm:
  - **Step 1:** Remove common suffixes (e.g., "ing," "ed," "s").
  - **Step 2:** Apply transformations to handle plural forms and past tense.
  - **Step 3:** Further reductions to handle derivational suffixes.
  - **Step 4:** Final cleanup to ensure the word is in its simplest form.
- **Example:** Given the words: "running," "runner," "runs"
  - Porter Stemmer reduces these to: "run," "run," "run"

Examples and Algorithms:

Indexing Example

Consider the documents:

- Doc 1: "Cats are great pets."
- Doc 2: "Dogs are great pets."
- Doc 3: "Cats and dogs can be great friends."

Inverted Index Creation:

1. **Tokenization:**
   - Doc 1: ["cats", "are", "great", "pets"]
   - Doc 2: ["dogs", "are", "great", "pets"]
   - Doc 3: ["cats", "and", "dogs", "can", "be", "great", "friends"]
2. **Normalization:**
   - Convert all tokens to lowercase (already lowercase here).

Index Creation:
json

```json
{
  "cats": [1, 3],
  "are": [1, 2],
  "great": [1, 2, 3],
  "pets": [1, 2],
  "dogs": [2, 3],
  "and": [3],
  "can": [3],
  "be": [3],
  "friends": [3]
}
```

3.

Stop Words Elimination Example

Given the document: "Cats are great pets and dogs are too"

- Stop words list: ["are", "and", "too"]
- After removal: "Cats great pets dogs"

Stemming Example

Given the words: "connecting," "connected," "connection"

- Porter Stemmer reduces these to: "connect," "connect," "connect"

Summary

- **Indexing** enables efficient search by creating structures like inverted indices that map terms to documents.
- **Stop words elimination** improves relevance by removing common words that do not add meaningful value.
- **Stemming** enhances retrieval by reducing words to their base form, thus treating related words as equivalent.

4	Explain basic Information Retrieval process with a neat diagram. State and explain the importance of Zip's law related to word distribution in NLP.	[5+5] des-3 eg-2	CO5	L2

The Information Retrieval (IR) process involves several steps to fetch relevant information from a large collection of data based on a user's query. Here's a step-by-step explanation along with a diagram to illustrate the process.

Steps in the Information Retrieval Process:

1. **Document Collection**: The collection of documents that will be searched. These documents can be any form of text, such as web pages, articles, books, etc.
2. **Preprocessing**: This step involves preparing the text for indexing and retrieval. It includes several sub-steps:
   - **Tokenization**: Splitting text into individual words or tokens.
   - **Stop Words Removal**: Eliminating common words that do not carry significant meaning (e.g., "the", "is", "in").
   - **Stemming/Lemmatization**: Reducing words to their base or root form.
3. **Indexing**: Creating an index to enable efficient search. An inverted index is commonly used, which maps each term to the documents in which it appears.
4. **Query Processing**: Similar to document preprocessing, the user's query is processed to prepare it for searching.
   - **Tokenization**: Splitting the query into individual terms.
   - **Stop Words Removal**: Removing common words.
   - **Stemming/Lemmatization**: Reducing query terms to their base form.
5. **Searching**: Matching the processed query against the index to retrieve relevant documents.
6. **Ranking**: Ordering the retrieved documents based on relevance to the query. This can be done using various ranking algorithms, such as TF-IDF (Term Frequency-Inverse Document Frequency) or BM25.
7. **Results Presentation**: Displaying the ranked list of documents to the user.

```
| Document      |            | Preprocessing|                | Indexing     |
| Collection    + ----->     | (Tokenization|                | (Inverted    |
| (Text Data)   |            | Stop Words   |   ----->       | Index)       |
|               |            | Removal,     |                |              |
|               |            | Stemming)    |                |              |

                                    |
                                    v
                             |   Index      |
                                    ^
                                    |

| User Query    |                 | Query       |          | Searching    |
| (Text Query)  +   ----->        | Processing  |----->    | (Match Query |
|               |                 | (Tokenization, |       | against Index)|
|               |                 | Stop Words  |          |              |
|               |                 | Removal,    |          |              |
|               |                 | Stemming)   |          |              |

                                    |
                                    v
                              | Ranking     |
                              | (TF-IDF,    |
                              | BM25)       |
                                    v
                              | Results     |
                              | Presentation |
```

<table>
<tr>
<td></td>
<td>

Zipf’s Law in NLP

Zipf’s Law states that in a given corpus of natural language, the frequency of any word is inversely proportional to its rank in the frequency table. Specifically, the second most frequent word occurs approximately half as often as the most frequent word, the third most frequent word occurs about one-third as often, and so on.

Formal Expression of Zipf’s Law:

f(r)≈Craf(r) \approx \frac{C}{r^a}f(r)≈raC

f(r)f(r)f(r): Frequency of the word with rank rrr
CCC: Constant
rrr: Rank of the word
aaa: Exponent close to 1 (typically around 1)

Importance of Zipf’s Law in NLP:

Understanding Vocabulary Distribution: Zipf’s Law helps in understanding the distribution of words in a language, which is crucial for tasks like corpus analysis, language modeling, and lexicon building.
Efficient Indexing: By recognizing that a small number of words appear very frequently while the majority appear rarely, IR systems can optimize indexing and storage. For example, stop words can be identified and handled differently to improve search efficiency.
Compression and Storage: Zipf’s Law is used in text compression algorithms because it reveals patterns in word frequency that can be exploited to reduce the size of stored text data.
Improving Search Performance: Understanding word frequency distribution helps in designing better ranking algorithms. For instance, frequent words might be given less weight in relevance scoring.
Handling Rare Words: Zipf’s Law indicates that most words in a large corpus are rare. Techniques like smoothing in language models are used to address the issue of rare words.
Resource Allocation: Helps in allocating computational resources effectively by focusing more on processing frequent terms that contribute most to the information content.

Example:

Consider a corpus with the following word frequencies:

"the": 5000 times (rank 1)
"is": 2500 times (rank 2)
"in": 1666 times (rank 3)
"and": 1250 times (rank 4)
"of": 1000 times (rank 5)

According to Zipf's Law, the word ranked 2 should appear about half as frequently as the word ranked 1, the word ranked 3 about one-third as frequently, and so on. This distribution helps in understanding the redundancy and importance of different words in text processing tasks.

</td>
<td>des-3
eg-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>

Explain the Cluster and fuzzy models of information retrieval systems with suitable examples.

The cluster model attempts to reduce the number of matches during retrieval.

The cluster hypothesis that explains why clustering could prove efficient in IR states that

“Closely associated documents tend to be relevant to the same clusters.”

The hypothesis suggests that closely associated documents are likely to be retrieved together.

By forming groups or classes or clusters of related documents, the search time reduces considerably.

Instead of matching a query with every document in the collection, it is matched with representatives of the cluster (class), and only documents from a class whose representative is close to query, are considered for individual match.

</td>
<td>[10]
Desc-5
examples-5</td>
<td>CO5</td>
<td>L2</td>
</tr>
</table>

Clustering is applied on terms instead of documents. Terms can be grouped to form classes of co-occurrence terms.

A number of methods are used to group documents. One of the method is based on similarity matrix.

Cluster Generation method based on Similarity Matrix

Let D = { $d_1, d_2, d_3, \ldots .. d_m$ } be set of documents.

Let E = $(e_{ij})_{n,n}$ be the similarity matrix.

The element $E_{i,j}$ in this matrix, denotes a similarity between document d_i and d_j.

Let T be the threshold value.

Any pair of documents d_i and d_j (i != j) whose similarity measure exceeds threshold (e_{ij} >= T) is grouped to form a cluster.

The remaining documents form a single cluster.

The set of clusters thus obtained is

$$C = \{ C_1, C_2, \ldots, C_k, \ldots, C_p \}$$

A representative vector of each cluster is constructed by computing the centroid of the document vectors belonging to that class.

Representation vector for a cluster C_k is $r_k = \{ a_{1k}, a_{2k}, \ldots, a_{ik}, \ldots, a_{mk} \}$

An element a_{ik} in this vector is computed as

$$a_{ik} = \frac{\sum_{d_j \in C_k} a_{ij}}{|C_k|}$$

Where a_{ij} is weight of the term t_i, of the document d_j, in cluster C_k.

During retrieval, the query is compared with the cluster vectors

$$(r_1, r_2, \ldots, r_k, \ldots, r_p)$$

This comparison is carried out by computing the similarity between the query vector q and the representative vector r, as

$$s_{ik} = \sum_{i=1}^{m} a_{ik} q_i, \quad k = 1, 2, \ldots, p$$

A cluster C, whose similarity s, exceeds a threshold is returned and the search proceeds in that cluster.

Example 9.4

Let

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

be the term-by-document matrix. The similarity matrix corresponding to these documents is

$$\begin{array}{lll} 1.0 & & \\ 0.9 & 1.0 & \\ 0.4 & 0.4 & 1.0 \end{array}$$

Using a threshold of 0.7, we get the following two clusters:

$C_1 = \{d_1, d_2\}$
$C_2 = \{d_3\}$

The cluster vectors (representatives) for C_1 and C_2 are

$r_1 = (1\ 0.5\ 1\ 0\ 1)$
$r_2 = (0\ 0\ \ 1\ 1\ 0)$

Retrieval is performed by matching the query vector with r_1 and r_2.

- In the **fuzzy model**, the document is represented as a fuzzy set of terms, i.e., a set of pairs **[t_i ,μ (t_i)]** Where μ is the membership function.

- The membership function assigns to each term of the document a numeric membership degree.

- The membership degree expresses the significance of term to the information contained in the document.

- Significance values (weights) are assigned based on the number of occurrences of the term in the document and in the entire document collection.

- Each document in the collection -

  $D = \{ d_1, d_2, \ldots, d_j, \ldots, d_n \}$

  Can thus be represented as a vector of term weights,

  $(w_{ij}, w_{2j}, w_{3j}, \ldots, w_{ij}, \ldots, w_{mj})^t$

Where w_{ij} is the degree to which term t_i belongs to document d_j.

- Each term in the document is considered a representative of a subject area and wij is the membership function of document dj to the subject area represented by term ti.

- Each term ti is represented by a fuzzy set fi in the domain of documents given by

  $f_i = \{ (d_j, w_{ij}) \} \mid i = 1, \ldots, m;\ j = 1, \ldots, n$

- This weighted representation makes it possible to rank the retrieved documents in decreasing order of their relevance to the user's query.

- Queries are Boolean queries.

- For each term that appears in the query, a set of documents is retrieved.

- Fuzzy set operations are then applied to obtain the desired result.

Single-term Query:

	• For a single-term query $q=t_q$, those documents from the fuzzy set $f_q =\{(d_j, w_{iq})\}$, are retrieved for which wiq exceeds a given threshold. The threshold may also be zero. **AND Query**: • For an AND query $q = t_{q1} \wedge t_{q2}$, the fuzzy sets fq1 and fq2 are obtained and then, their intersection is obtained, using the fuzzy intersection operator $f_{q1} \wedge f_{q2} = \min \{ (d_j, w_{iq1}), (d_j, w_{iq}) \}$ The documents in this set are returned. **OR Query:** • For an OR query $q = t_{q1} V t_{q2}$, the union of fuzzy sets fq1 and fa2 is computed to retrieve documents as follows - $f_{q1} V f_{q2} = \max \{ (d_j, w_{iq1}), (d_j, w_{iq}) \}$ Consider the following 3 documents: d_1 = { information, retrieval, query } d_2 = { retrieval, query, model } d_3 = { information, retrieval } Where the set of terms used to represent documents is T = { information, model, query, retrieval } Fuzzy set for terms $f_1 = \{ (d_1, 1/3), (d_2, 0), (d_3, 1/2) \}$ → t_1 = information $f_2 = \{ (d_1, 0), (d_2, 1/3), (d_3, 0) \}$ → t_2 = model $f_3 = \{ (d_1, 1/3), (d_2, 1/3), (d_3, 0) \}$ → t_3 = query $f_4 = \{ (d_1, 1/3), (d_2, 1/3), (d_3, 1/2) \}$ → t_4 = retrieval If the query is $q = t_2 \wedge t_4$, then document d_2 is returned.			
6 a)	Define term weighting. Consider a document represented by the three terms {tornado, swirl, wind} with the raw tf 4, 1 and 1 respectively. In a collection of 100 documents, 15 documents contain the term tornado, 20 contain swirl and 40 contain wind. Find the idf and the term weight of the three terms. idf - tornado -> log(n / ni) = log (100 / 15) = 0.824 Weight - tornado -> tf x idf = 4 * 0.824 = 3.296 idf - swirl -> log(n / ni) = log (100 / 20) = 0.699 Weight - tornado -> tf x idf = 1 * 0.699 = 0.699 idf - wind -> log(n / ni) = log (100 / 40) = 0.398 Weight - tornado -> tf x idf = 1 * 0.398 = 0.398	[5] Desc-3 calculator-2	CO5	L3

Term	Frequency (tf)	Document frequency (n_i)	idf [$\log(n/n_i)$]	Weight (tf × idf)
Tornado	4	15	0.824	0.296
Swirl	1	20	0.699	0.699
Wind	1	40	0.398	0.389

b) Explain WordNet and FrameNet with suitable examples and write the hypernym chain for 'RIVER' extracted from WordNet 2.0 — [5] desc-3 eg:2 | CO5 | L3

WordNet:

WordNet is a large lexical database for the English language. Inspired by psycholinguistic theories, it was developed and is being maintained at the Cognitive Science Laboratory, Princeton University, under the direction of George A. Miller.

WordNet consists of three databases

- One for nouns
- One for verbs
- One for both adjectives and adverbs

Information is organized into sets of synonymous words called **synsets**, each representing one base concept. The synsets are linked to each other by means of lexical and semantic relations. Lexical relations occur between word forms (senses) and semantic relations between word meanings. These relations include synonymy, hypernymy / hyponymy, antonymy, meronymy / holonymy, troponymy, etc. A word may appear in more than one synset and in more than one part of speech. The meaning of the word is called sense. WordNet lists all senses of a word, each sense belonging to a different synset.

WordNet's sense entries consist of set synonyms and a gloss. A gloss consists of a dictionary-style definition and examples demonstrating the use of a synset in a sentence. Glosses help differentiate meanings.

The Figure below shows Noun relations in WordNet

Nouns and verbs are organized into hierarchies based on the hypernymy/hyponymy relation.

Relation	*Definition*	*Example*
Hypernym	From concepts to super-ordinates	oak→ tree
Hyponym	From concepts to subtypes	oak→ white oak
Meronym	From wholes to parts	tree → trunk
Holonym	From parts to wholes	trunk → tree
Antonym	Opposites	victory → defeat

Figure 12.2 Noun relations in WordNet

Relation	Definition	Example
Hypernym	From events to super-ordinate events	wander→ travel
Troponym	From events to their subtypes	walk → stroll
Entails	From events to the events they entail	snore → sleep
Antonym	Opposites	increase → decrease

Figure 12.3 Verb relations in WordNet

Relation	Definition	Example
Antonym (adjective)	Opposite	heavy → light
Antonym (adverb)	Opposite	quickly → slowly

Figure 12.4 Adjective and adverb relations in WordNet

WordNet is freely and publicly available for download from http://wordnet.princeton.edu/obtain. WordNet for other languages have been developed, for example, **EuroWordNet** and **Hindi WordNet.**

EuroWordNet covers European languages, including English, Dutch, Spanish, Italian, German, French, Czech, and Estonian.

Hindi WordNet has been developed by CFILT (Resource Center for Indian Language Technology Solutions), IIT Bombay. Its database consists of more than 26208 synsets and 56928 Hindi words. It is organized using the same principles as English WordNet but include some Hindi-specific relations - causative relations. Hindi WordNet can be obtained from the URL http://www.cfilt.iitb.ac.in/wordnet/webhwn/

CFLIT has also developed a **Marathi WordNet** http://www.cfilt.iitb.ac.in/wordnet/webmwn/wn.php

Applications of WordNet:

- **Concept identifications in Natural language**
  WordNet can be used to identify concepts pertaining to a term, to suit them to the full semantic richness.
- **Word sense disambiguation**
  It offers
  - sense definitions of words
  - identifies synsets of synonyms
  - Defines a number of semantic relations
- **Automatic Query Expansion**
  WordNet semantic relations can be used to expand queries so that the search for a document is not confined to the pattern-matching of query terms, but also covers synonyms.
- **Document structuring and categorization**
  The semantic information extracted from WordNet has been used for text categorization.
- **Document summarization**
  The approach presented by Barzilay and Elhadad uses information from WordNet to compute lexical chains.

FrameNet:

- FrameNet is a large database of semantically annotated English sentences.
- It is based on principles of frame semantics.
- It defines a tagset of semantic roles called the **frame element**.
- Sentences from the British National Corpus are tagged with these frame elements.
- The basic philosophy involved is that each word evokes a particular situation with particular participants.
- FrameNet aims at capturing these situations through the case-frame representation of words (verbs, adjectives, and nouns).
- The word that invokes a frame is called the target word or predicate, and the participant entities are defined using semantic roles, which are called frame elements.
- The FrameNet ontology can be viewed as a semantic-level representation of the predicate-argument structure.
- Each frame contains a main lexical item as a predicate and associated frame-specific semantic roles, such as AUTHORITIES, TIME, AND SUSPECT in the ARREST frame, called frame elements.
- Example: The sentence below is annotated with semantic roles AUTHORITIES AND SUSPECT

  [$_{\text{Authorities}}$ The police] nabbed [$_{\text{suspect}}$ the snatcher].

- The COMMUNICATION frame has the semantic roles ADDRESSEE, COMMUNICATOR, TOPIC, and MEDIUM.
- A JUDGEMENT frame contains roles such as a JUDGE, EVALUEE, and REASON.
- Example:

  [$_{\text{judge}}$ She] [$_{\text{Evaluee}}$ blames the police] [$_{\text{Reason}}$ for failing to provide enough protection].
- A frame may inherit roles from another frame. Eg., a STATEMENT frame may inherit from a COMMUNICATION frame, it contains roles such as SPEAKER, ADDRESSEE, and MESSAGE.
- Example:

  [$_{\text{Speaker}}$ She] told [$_{\text{Addressee}}$ me] [$_{\text{Message}}$ 'I'll return by 7:00 pm today'].

Applications of FrameNet:

FrameNet data can be used for

1. Automatic semantic parsing
2. Information extraction
3. Question answering system
4. Information retrieval
5. Machine translation
6. Text summarization
7. Word sense disambiguation

	1 sense of 'river' Sense 1 river — (a large natural stream of water (larger than a creek); 'the river was navigable for 50 miles') => stream, watercourse — (a natural body of running water flowing on or under the earth) => body of water, water — (the part of the earth's surface covered with water (such as a river or lake or ocean); 'they invaded our territorial waters'; 'they were sitting by the water's edge') => thing — (a separate and self-contained entity) => entity — (that which is perceived or known or inferred to have its own distinct existence (living or nonliving)) **Figure 12.5** Hypernym chain for 'river'			

CI CCI HOD-AIML