ARTIFICIAL INTELLIGENCE

L T P C

BCSE306L - 3 0 0 3

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Dr. S M SATAPATHY

Associate Professor Sr.,
School of Computer Science and Engineering,
VIT Vellore, TN, India – 632 014.

Module – 7

COMMUNICATING, PERCEIVING AND ACTING

1. Language Models
2. Information Retrieval
3. Information Extraction
4. Image Formation
5. Object Recognition

NLP

NLP :: Introduction

- Human speech ~100,000 years
- Writing ~ 7,000 years
- Huge numbers of pages of information on the Web, almost all of it in natural language
- For knowledge acquisition an agent needs to understand (partially) the ambiguous, messy languages that humans use
- Specific information-seeking tasks:
  - text classification
  - information retrieval
  - information extraction
- Requires to use language models, which predict the probability distribution of language expressions.

NLP :: Introduction

- What is NLP?
  - Fundamental goal: analyze and process human language, broadly, robustly, accurately…
- End systems that we want to build:
  - Ambitious: speech recognition, machine translation, information extraction, dialog interfaces, question answering…
  - Modest: spelling correction, text categorization…
- Problem: Ambiguities:
  - Enraged Cow Injures Farmer With Ax
  - Hospitals Are Sued by 7 Foot Doctors
  - Juvenile Court to Try Shooting Defendant
  - Stolen Painting Found by Tree
  - Kids Make Nutritious Snacks

Language Models

- Formal languages, such as the programming languages Java or Python, have precisely defined language models
- A language can be defined as a set of strings;
  - "print(2 + 2)" is a legal program in the language Python, whereas "2)+(2 print" is not
- Infinitely many legal programs: they cannot be enumerated
  - they are specified by a set of rules called a grammar
- Formal languages also have rules that define the meaning of semantics of a program
  - the rules say that the "meaning" of "2 + 2" is 4, and
  - the meaning of "1/0" is that an error is signalled
- Natural languages, such as English or Spanish, cannot be characterized as a definitive set of sentences

Characteristics of natural languages

"Not to be invited is sad"

- is a sentence of English, but what about the grammaticality of

"To be not invited is sad"

- A natural language model = probability distribution over sentences rather than a definitive set
- Natural languages are also ambiguous
  - "He saw her duck" can mean either that he saw a waterfowl belonging to her, or that he saw her move to evade something
  - There is no single meaning for a sentence, but rather of a probability distribution over possible meanings
- Natural languages are very large
  - hence difficult to deal with, and constantly changing
  - Thus, our language models are, at best, an approximation

Characteristics of natural languages

Example:

- Imagine you ask an NLM to complete the sentence "The cat sat on the..." Here's how it might work:
  - The NLM analyzes its vast database of text and finds sentences containing "The cat sat on the...".
  - It identifies the most frequent following words: "mat" (very common), "rug" (common), "porch" (less common), and maybe even more exotic options like "zebra" (very rare).
  - The NLM wouldn't give you a single answer like "mat." Instead, it would assign probabilities to each completion. "Mat" would have the highest probability, followed by "rug" and so on.

Characteristics of natural languages

Example:

- Imagine you ask an NLM to translate the sentence "I'm feeling hungry" into French.
- A traditional rule-based translation system might have a single pre-programmed translation like "J'ai faim" (one correct answer).
- An NLM, however, would approach it differently. It would analyze its database of translated sentences and identify various ways to express "I'm feeling hungry" in French. It might find options like:
  - "J'ai faim" (most common)
  - "J'ai un creux dans l'estomac" (literally "I have a hole in my stomach," less common)
  - "Je commence à avoir faim" (literally "I'm starting to get hungry," slightly different nuance)

Characteristics of natural languages

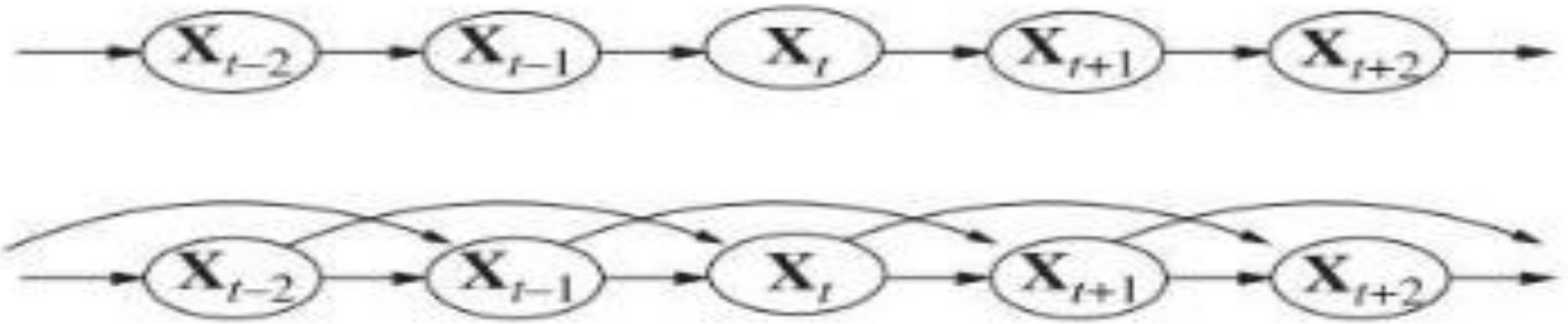
Example:

- The NLM wouldn't simply pick one translation. Instead, it would assign probabilities to each option based on how frequently it appears in the data. "J'ai faim" would likely have the highest probability, followed by the others.
- This probabilistic approach allows the NLM to consider the context and nuances of language. Depending on the situation, "Je commence à avoir faim" might be a more suitable choice (e.g., if you're politely informing someone you're getting hungry).

N-gram character models

- Ultimately, a written text is composed of characters —
  - letters, digits, punctuation, and spaces in English
- One of the simplest language models is a probability distribution over sequences of Characters
- A sequence of written symbols of length n is called an $\boldsymbol{n}$-gram
- A model of the probability distribution of n-letter sequences is called an n-gram model
- We can also have n-gram models over sequences of words, syllables, or other units.

Markov chain



- A transition model specifies a probability distribution over the latest state variables, given the previous values, that is,

$$P(X_t \mid X_{0:t-1})$$

- Problem: the set $X_{0:t-1}$ is unbounded in size as t increases
- Solve by making a Markov assumption
  - the current state depends on only a finite fixed number of previous states

Markov chain

- Markov processes or Markov chains first studied in depth by the Russian statistician Andrei Markov (1856–1922)
- The simplest model is the first-order Markov process, in which the current state depends only on the previous state and not on any earlier states
  - i.e., a state provides enough information to make the future conditionally independent of the past, and we have

$$P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$$

- In a first-order Markov process, the transition model is the conditional distribution $P(X_t \mid X_{t-1})$
- The transition model for a second-order Markov process is the conditional distribution, $P(X_t \mid X_{t-2}, X_{t-1})$

Markov chain

- An n-gram model is defined as a Markov chain of order $n - 1$
- In a Markov chain the probability of character c_i depends only on the immediately preceding characters, not on any other characters
- So in a trigram model (Markov chain of order 2) we have

$$P(C_i \mid C_{1:i-1}) = P(C_i \mid C_{i-2:i-1})$$

- We define the probability of a sequence of characters $P(c_{1:N})$ under the trigram model by first factoring with the chain rule and then using the Markov assumption:

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

- For a trigram model in a language with 100 characters, $P(C_i \mid C_{i-2:i-1})$ has a million (100^3) entries.
- A body of text is called corpus, from the Latin word for body

Model evaluation

- There are many possible n-gram models - unigram, bigram, trigram, interpolated smoothing with different values of λ
- We can evaluate a model with cross-validation
  - Split the corpus into a training corpus and a validation corpus
  - Determine the parameters of the model from the training data
  - Then evaluate the model on the validation corpus
- The evaluation can be a task-specific metric
  - E.g., measuring accuracy on language identification
- Alternatively we can have a task-independent model of language quality:
  - calculate the probability assigned to the validation corpus by the model;
  - the higher the probability the better

N-gram word models

- In n-gram models over words all the same mechanisms apply as in character models. The vocabulary is larger
- There are only about 100 characters in most languages, and sometimes we build character models that are even more restrictive, e.g.,
  - by treating “A” and “a” as the same symbol or
  - by treating all punctuation as the same symbol
- With word models we have tens of thousands of symbols, and sometimes millions
- It is not clear what constitutes a word
  - English: a sequence of letters surrounded by spaces is a word
  - Chinese: words are not separated by spaces
- Even in English many decisions must be made to have a clear policy on word boundaries: how many words are in “ne’er-do-well”? Or in “(Tel:1-800-960-5660x123)”?

Unknown words

- Word n-gram models need to deal with out of vocabulary words.
- With word models there is always the chance of a new word that was not seen in the training corpus
- Add just one new word to the vocabulary:

  <UNK> , standing for the unknown word
- We can estimate n-gram counts for UNK: go through the training corpus, and the first time any individual word appears it is previously unknown, so replace it with UNK
- All subsequent appearances of the word remain unchanged
- Then compute n-gram counts as usual, treating UNK just like any other word
- When an unknown word appears in a test set, we look up its probability under UNK

Unknown words

- Build models over the words in the AIMA textbook and then randomly sample sequences of words from the them
  - Unigram: logical are as are confusion a may right tries agent goal the was…
  - Bigram: systems are very similar computational approach would be represented…
  - Trigram: planning and scheduling are integrated the success of naive Bayes model…
- A different way of describing the probability of a sequence is with a measure called perplexity, defined as

$$\text{Perplexity}(c_{1:N}) = P(c_{1:N})^{(-1/N)}$$

Unknown words

- Perplexity can be thought of as the reciprocal of probability, normalized by sequence length.
- It can also be thought of as the weighted average branching factor of a model.
- The unigram model (perplexity 891) is a poor approximation of either English or the content of an AI textbook. The bigram (perplexity 142) and trigram (perplexity 91) models are much better.

Perplexity Example - Unigram

- Our sentence has five words: "The", "cat", "sat", "on", "mat".
- P("The") = 0.1 (assuming "The" is a very common word)
- P("cat") = 0.05
- P("sat") = 0.02
- P("on") = 0.03
- P("mat") = 0.04
- Perplexity = 1 / (Product of all word probabilities)

  Perplexity = 1 / (0.1 * 0.05 * 0.02 * 0.03 * 0.04) => Perplexity ≈ 500
- Interpretation:
  - A high perplexity value (500 in this example) indicates that the unigram model is not very good at predicting the entire sequence of words.
  - This is because the unigram model ignores the order of words and treats each word independently.

Perplexity Example - Bigram

- Our sentence has five words, but for bigram analysis, we consider word pairs:
  - Pair 1: "The cat"
  - Pair 2: "cat sat"
  - Pair 3: "sat on"
  - Pair 4: "on the"
  - Pair 5: "the mat"
- P("sat" following "The cat") = 0.3 (assuming "cat" often precedes "sat")
- P("on" following "cat sat") = 0.2
- P("the" following "sat on") = 0.1 (unrealistic for this sentence, but illustrates the concept)
- P("mat" following "on the") = 0.8 (assuming "the" often precedes "mat")
- We won't calculate P("The") as bigrams don't predict the first word.

Perplexity Example - Bigram

- Perplexity = 1 / (Product of all bigram probabilities)

Perplexity ≈ 1 / (0.3 * 0.2 * 0.1 * 0.8) => Perplexity ≈ 16.67 (assuming all probabilities are accurate)

- Interpretation:
  - Compared to the unigram model (perplexity of 500), the bigram model has a significantly lower perplexity (16.67). This indicates a better ability to predict the sequence of words.
  - By considering the previous word, the bigram model can make more informed predictions.
  - For example, after seeing "The cat," the model knows "sat" is a more likely following word than "blue" because "cat sat" is a more frequent bigram in the training data.

Perplexity Example - Trigram

- Our sentence has five words. Here, we analyze triplets (three consecutive words):
  - Triplet 1: "The cat sat"
  - Triplet 2: "cat sat on"
  - Triplet 3: "sat on the"
  - Triplet 4: "on the mat"
- P("sat" following "The cat") = 0.8 (assuming "The cat sat" is a common phrase)
- P("on" following "cat sat") = 0.7 (assuming "cat sat on" is a frequent sequence)
- P("the" following "sat on") = 0.05 (realistic in this context, "the" might not always follow "sat on")
- P("mat" following "on the") = 0.9 (assuming "on the mat" is a common ending)

Perplexity Example - Trigram

- Perplexity = 1 / (Product of all trigram probabilities)

Perplexity ≈ 1 / (0.8 * 0.7 * 0.05 * 0.9) => Perplexity ≈ 2.08 (assuming all probabilities are accurate)

- Interpretation:
  - The trigram model has a significantly lower perplexity (2.08) compared to both the unigram (500) and bigram (16.67) models. This indicates a superior ability to predict the sequence of words due to the larger context considered.
  - By analyzing the previous two words, the trigram model can make even more refined predictions. For example, after seeing "The cat sat," it knows "on" is much more likely than "blue" because "cat sat on" is a more frequent trigram in the training data.

Text Classification

- Text Classification (Categorization): Assigns a text to a predefined category (e.g., language identification, genre classification, sentiment analysis).
- Spam Detection: Classifies emails as spam or not-spam (also called "ham").
  - Supervised Learning: Uses training data (spam folder & inbox) to build a model.
- Spam Features:
  - Word n-grams (e.g., "for cheap", "You can buy").
  - Character-level features (uppercase letters, punctuation in words).

Text Classification

Two Approaches to Classification:

- Language Modeling Approach:
  - This approach involves training separate n-gram language models for each category (spam and ham).
  - An n-gram model predicts the probability of a sequence of words (n-grams) occurring in a text.
  - In this case, we train one model on spam emails (P(Message | spam)) and another on regular emails (P(Message | ham)).
- Classification happens by applying Bayes' rule:
  - We calculate the probability of a new message belonging to each category (spam and ham) using the trained models and prior probabilities (P(spam) and P(ham)).
  - The message is classified into the category with the highest resulting probability.

Text Classification

Two Approaches to Classification:

- Machine Learning with Feature Engineering:
  - This approach focuses on designing informative features specifically for spam detection. These features go beyond simple n-grams and aim to capture the characteristics of spam emails.
  - Examples of features could be:
    - Presence of URLs or specific keywords often used in spam.
    - Blacklisted email addresses or domains.
    - Analysis of email headers (e.g., sender information).
    - Character-level features like the number of uppercase letters or punctuation embedded in words.
  - These features are then used to train a machine learning model (e.g., decision tree, support vector machine) to classify new emails as spam or ham.

Text Classification

Classification by data compression

- A lossless compression algorithm takes a sequence of symbols, detects repeated patterns in it, and writes a description of the sequence that is more compact than the original.
- For example, the text “0.14285714285714285” might be compressed to “0.[142857]*3.”
- Compression algorithms work by building dictionaries of subsequences of the text, and then referring to entries in the dictionary.
- The example here had only one dictionary entry, “142857.”or ham.
- In effect, compression algorithms are creating a language model.
- The Lempel-Ziv-Welch (LZW) algorithm in particular directly models a maximum-entropy probability distribution.

Information Retrieval

- Information Retrieval (IR): Finding relevant documents for a user's information need.
- Example: Search engines finding webpages related to your query (e.g., [AI book]).
- Key Components of an IR System:
  - Corpus: Collection of documents (paragraphs, pages, or multi-page texts).
  - Query Language: How users specify their information need (keywords, phrases, Boolean operators like AND/OR, proximity operators like NEAR).
  - Result Set: Subset of documents deemed relevant to the user's query.
  - Presentation: Ranked list, summaries, or visualizations of the results.

Information Retrieval - Boolean Keyword Model

- This is the simplest model and was commonly used in early information retrieval systems.
- It treats each word in the document collection as a binary feature:
  - True: If the word appears in the document.
  - False: If the word doesn't appear in the document.
- Users express their information needs using Boolean operators like AND, OR, and NOT to combine keywords. Example Query: AI AND book (retrieves documents containing both "AI" and "book").
- Limitations:
  - Doesn't consider word order or proximity. Documents with "artificial" and "intelligence" mentioned separately might be retrieved even though they're not about AI.
  - Doesn't account for word importance or meaning. All words are treated equally.

Information Retrieval - Bag-of-Words(BoW) Model

- This model represents a document as an unordered collection of words (the bag). Each word's frequency (how many times it appears) can be used as a feature for the document.
- Documents are compared based on their word frequency vectors.
- Simpler and more efficient to compute compared to the Boolean model.
  - Example: A document about AI might have a high frequency for words like "artificial intelligence", "machine learning", etc.
- Limitations:
  - Ignores word order and context. Documents with similar word frequencies but different meanings might be considered relevant.
  - Doesn't capture word relationships or semantics.

Information Retrieval - Complete Representation Model

- This is a more advanced model that aims to capture a richer representation of documents and user queries.
- It goes beyond simple keyword presence or frequency and tries to incorporate additional information.
  - Examples of features used in complete representation models:
- Word stemming/lemmatization (reducing words to their root form).
- Term Frequency-Inverse Document Frequency (TF-IDF): Weights words based on their importance within a document and across the document collection.
- Word co-occurrence: Captures how often words appear together, providing insights into semantic relationships.
- Positional information: Considers the order of words in a document.

Information Retrieval - Complete Representation Model

- Advantages:
  - Provides a more nuanced understanding of documents and queries.
  - Improves retrieval accuracy by considering word importance, context, and relationships.

- Challenges:
  - Can be computationally expensive compared to simpler models.
  - Requires careful selection and weighting of features for optimal performance.

Information Retrieval

- BM25 Scoring Function: Rates document relevance for a given query (higher score = higher relevance).
- Factors Affecting Score:
  - Term Frequency (TF): How often a query term appears in a document (e.g., "farming" in a farming in Kansas document).
  - Inverse Document Frequency (IDF): How common a word is across documents (common words like "in" have lower IDF).
  - Document Length: Shorter documents mentioning all query terms are more relevant than long ones containing them incidentally.

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L})},$$

$$IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

Information Retrieval

where

- $|d_j|$ is the length of document d_j in words, and
- L is the average document length in the corpus.
- We have two parameters, k and b, that can be tuned by cross-validation; typical values are k = 2.0 and b = 0.75.
- IDF(q_i) is the inverse document frequency of word q_i

Information Retrieval

- IR System evaluation: Imagine that an IR system has returned a result set for a single query, for which we know which documents are and are not relevant, out of a corpus of 100 documents.

	In result set	Not in result set
Relevant	30	20
Not relevant	10	40

- Traditionally, there have been two measures used in the scoring: recall and precision.
- Precision measures the proportion of documents in the result set that are actually relevant. In our example, the precision is 30/(30 + 10)=.75. The false positive rate is 1 − .75=.25.
- Recall measures the proportion of all the relevant documents in the collection that are in the result set. In our example, recall is 30/(30 + 20)=.60. The false negative rate is 1 − .60=.40.

Information Retrieval

- In a very large document collection, such as the WorldWideWeb, recall is difficult to compute, because there is no easy way to examine every page on the Web for relevance.
- All we can do is either estimate recall by sampling or ignore recall completely and just judge precision.
- In the case of a Web search engine, there may be thousands of documents in the result set, so it makes more sense to measure precision for several different sizes, such as “P@10” (precision in the top 10 results) or “P@50,” rather than to estimate precision in the entire result set.
- A Summary of both measures is the F1 score, a single number that is the harmonic mean of precision and recall, 2PR/(P + R).

Information Retrieval

- IR refinement:
  - Stemming: Reduce words to their base form (e.g., "couches" to "couch"). Can improve recall (finding relevant documents) but might hurt precision (accuracy) due to ambiguity (e.g., "stocking").
  - Synonym Recognition: Account for synonyms like "sofa" for "couch." Improves recall but can affect precision if synonyms aren't perfect matches (e.g., "Tim Couch" vs. sofas).
  - Related Words: Consider related terms like "leather" or "modern" to confirm document relevance.
  - Query Expansion: Learn from user queries (e.g., "new sofa" followed by "new couch") to automatically expand future queries.

Information Retrieval

The PageRank & HITS algorithm:

function HITS(*query*) **returns** *pages* with hub and authority numbers

pages ← EXPAND-PAGES(RELEVANT-PAGES(*query*))
for each *p* **in** *pages* **do**
  p.AUTHORITY ← 1
  p.HUB ← 1
repeat until convergence **do**
  for each *p* **in** *pages* **do**
    p.AUTHORITY ← $\sum_i$ INLINK$_i$(*p*).HUB
    p.HUB ← $\sum_i$ OUTLINK$_i$(*p*).AUTHORITY
  NORMALIZE(*pages*)
return *pages*

Information Retrieval

Information Retrieval vs. Question Answering:

- Information Retrieval: Finds documents relevant to a query (topic, concept).
- Question Answering: Provides a short answer (sentence or phrase) to a specific question.
- Web-based Question Answering Systems:
  - Leverage the vast amount of information on the web to answer questions.
  - Focus on precision (finding a single correct answer) rather than recall (finding all possible answers).

Information Retrieval

Information Retrieval vs. Question Answering:

- Example System (ASKMSR):
  - Rephrases user questions into search engine queries (e.g., "Who killed Abraham Lincoln?" becomes "* killed Abraham Lincoln").
  - Analyzes snippets from search results, not full webpages.
  - Identifies frequent n-grams (word sequences) in the results.
  - Filters n-grams based on expected answer type (name, number, date, etc.) and removes parts of the question itself.
  - Returns the highest-scoring n-gram as the answer.

Information Extraction

- Information Extraction (IE): Automatically acquiring knowledge from text.
- Key Task: Finding specific objects (e.g., addresses, storms) and their details within text.
- A typical task is to extract instances of addresses from Web pages, with database fields for street, city, state, and zip code; or instances of storms from weather reports, with fields for temperature, wind speed, and precipitation.
- Limited vs. General Domains:
  - Limited domains (e.g., web addresses) allow high accuracy with simpler models.
  - General domains require complex linguistic models and learning techniques

Information Extraction

- Attribute-Based Information Extraction (ABIE): Extracts attributes of a single object from text.
  - Example: Extracting product details (Manufacturer, Model, Price) from a product description.
- Using Templates for Attribute Values:
  - Templates define patterns to match attribute values in text.
  - Finite-state automata (like regular expressions) are used to build templates.
- Regular Expressions (Regex): Used to define patterns for matching text (e.g., prices in this case). Examples:

`[0-9]`	matches any digit from 0 to 9
`[0-9]+`	matches one or more digits
`[.][0-9][0-9]`	matches a period followed by two digits
`([.][0-9][0-9])?`	matches a period followed by two digits, or nothing
`[$][0-9]+([.][0-9][0-9])?`	matches $249.99 or $1.23 or $1000000 or . . .

Information Extraction

- Template Parts:
  - Prefix Regex: Text appearing before the attribute value (e.g., "price:").
  - Target Regex: The actual pattern matching the attribute value (e.g., price format).
  - Postfix Regex: Text appearing after the attribute value (can be empty).
- Handling Multiple Matches:
  - Priority Templates: Use templates with different prefixes in a set order (e.g., "our price:" first, then "price:").
  - Selection Strategies: Choose the best match among multiple (e.g., lowest price within a range).pen_sparktunesharemore_vert

Information Extraction

- One step up from attribute-based extraction systems are relational extraction systems, which deal with multiple objects and the relations among them.
- Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.

$$e \in JointVentures \wedge Product(e, \text{“}golf\ clubs\text{”}) \wedge Date(e, \text{“}Friday\text{”}) \\ \wedge Member(e, \text{“}Bridgestone\ Sports\ Co\text{”}) \wedge Member(e, \text{“}a\ local\ concern\text{”}) \\ \wedge Member(e, \text{“}a\ Japanese\ trading\ house\text{”})\ .$$

- A relational extraction system can be built as a series of cascaded finite-state transducers.

Information Extraction

```
1 NG: Bridgestone Sports Co.
2 VG: said
3 NG: Friday
4 NG: it
5 VG: had set up
6 NG: a joint venture
7 PR: in
8 NG: Taiwan
9 PR: with
10 NG: a local concern
11 CJ: and
12 NG: a Japanese trading house
13 VG: to produce
14 NG: golf clubs
15 VG: to be shipped
16 PR: to
17 NG: Japan
```

- Here NG means noun group, VG is verb group, PR is preposition, and CJ is conjunction.

Information Extraction

- Limitations of Finite-State Automata (FSA) for Information Extraction:
  - Noisy or varied input can be challenging for rule-based approaches.
  - Defining and prioritizing complex rules becomes difficult.
- Probabilistic Models for Information Extraction:
  - Offer an alternative to rule-based models.
  - Handle noisy or varied data more effectively.
- Hidden Markov Models (HMMs) for Information Extraction:
  - Model sequences with hidden states (e.g., attribute parts) and observations (words).
  - Two options: One large HMM for all attributes or separate HMMs for each.

Information Extraction

- Applying HMMs to Information Extraction:
  - Words are observations.
  - Hidden states represent attribute parts (target, prefix, postfix) or background.
  - HMMs are trained on data, not hand-crafted with templates.
- Advantages of HMMs over FSAs:
  - Probabilistic: Tolerate noise and missing elements, providing a confidence score.
  - Trainable: Learn from data, adapting to changing text formats.

Information Extraction

- Few other techniques for information extraction:
  - Conditional random fields for information extraction
  - Ontology extraction from large corpora
  - Automated template construction
  - Machine reading

Disclaimer

The material for the presentation has been compiled from various sources such as prescribed text books by Russell and Norvig and other tutorials and lecture notes. The information contained in this lecture/ presentation is for educational purpose only.

Thank You *for* Your Attention !