

19LINC-401


COMPUTATIONAL LINGUISTICS

Compiled by

Dr. P. Vijaya,

Assistant Professor

CAS in Linguistics, Annamalai University

ANNAMALAI  **UNIVERSITY**
CENTRE OF ADVANCED STUDY IN LINGUISTICS
ANNAMALAI NAGAR

2020

SYLLABUS

Learning objective:

- ❖ To enable the students to linguistically study the language as it used in the field of computational linguistics.
- ❖ To understand the computational knowledge.
- ❖ To provide students with an understanding of the social dimension of language and its implication for applied areas including language education and linguistics policy.
- ❖ To provide students with an language components with computation field.
- ❖ To familiarize students with the basic concepts and methods of computational linguistics.

Unit–I: Computational Phonetics and Phonemics

Introduction to computer: Types of Computer, generations of computer – Anatomy of computer. Articulatory phonetics vs. Acoustic phonetics: Prosodic features, speech signal processing, parameters and features of speech - Finite state implementation of phonological rules - Introduction to speech synthesis – text-to-speech system, speech recognition – speech-to-text system.

Unit–II: Computational Morphology and Syntax

Introduction to Morphology – morpheme: free Vs bound, Morphological Processing – Inflectional, Derivational and Compositional morphology – word structure, Morphological analysis – different approaches. Representation of morphological information: Finite –State Morphological parsing, Morphotactics, Natural Language Processing, Parsing – definition, Classification of parsing – Top-down vs. Bottom-up Parsing; Features and Augmented Grammars, Parsing with Features; Augmented Transition Networks; Generalized Feature systems and Unification Grammars- morphological recognizers, analyzers.

Introduction to Generalized Phrase Structure Grammar (GPSG), Definite Clause Grammar(DCG), Lexical Functional Grammar (LFG),Head-driven Phrase Structure Grammar(HPSG) and Tree Adjoining Grammar (TAG). Feature systems and Augmented Transition Networks.

Unit–III: Semantics and Knowledge Representation

Reference and compositionality, Functions and Predicate-Argument Structure; Meanings of referring expressions; Verifiability; Unambiguous Representations; Canonical Form; Inference and Variables; Expressiveness; Meaning Structure of Language; First Order Predicate Calculus; Elements of FOPC; the Semantics of FOPC; Variables and Quantifiers; Inferences.

Unit–IV: Computational Lexicography

Introduction to lexicography – Dictionary information – stages of dictionary preparation: data collection, entry selection, entry construction and entry arrangement. Role of computers in each stage, computer based dictionary making - Machine Readable Dictionary (MRD), Lexical resources, Role of language corpus in Lexicography, Electronic Dictionary (ED); Advantages of ED over conventional dictionary.

Unit–V: Application of Computational Linguistics

Machine Translation (MT) –different approaches; direct, interlingual, interlingual transfer – problems in lexical transfer – Computer Aided Learning / Teaching– role of computational linguistics in language teaching; Building Search Engines and Information retrieval system – Corpus Linguistics-Types of corpus Linguistics-role of corpus linguistics in teaching.

Unit–I

Computational Phonetics and Phonemics

Introduction to Computer

Computer is an electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.

Types of computers

- Based on Classifications there are three types of computer.
 1. Classification based on operations
 2. Classification based on size, speed and accuracy
 3. Classification based on designs

1. Classification of Computer based on operations

There are three types of Computers:

- I. Digital Computer
- II. Analog Computer
- III. Hybrid Computer

2. Based on size, speed and accuracy

There are four types of Computers:

- I. Supercomputer
- II. Mainframe Computer
- III. Minicomputer
- IV. Microcomputer

3. Based on Hardware designs

There are five types of Computers:

- I. General Purpose Computer
- II. Special Purpose Computer

- III. Machine Inbuilt computer
- IV. High Intelligence Computer
- V. Artificial Intelligence Computer

1. Classification of computer based on operations

There are three types of Computers:

i. Digital Computer

Digital **computer** does not measure the continuous data for continuous output.

ii. Analog Computer

An **analog computer** is a type of **computer** that uses the continuously changeable aspects of physical phenomena such as electrical, mechanical, or hydraulic quantities to model the problem being solved. ... **Analog computers** can have a very wide range of complexity.

iii. Hybrid Computer

The hybrid **computer** is the combination of analog and digital **computer** system.

2. Classification of Computer Based on size, speed and accuracy

There are four types of Computers:

i. Supercomputer

The most powerful computers in terms of performance and data processing are the Supercomputers. These are specialized and task specific computers used by large organizations. The supercomputers are very expensive and very large in size. It can be accommodated in large air-conditioned rooms; some super computers can span an entire building.

ii. Mainframe Computer

Mainframe Computers are used primarily by large organizations for critical applications; bulk data processing, such as census, industry and consumer statistics, enterprise resource planning; and transaction processing.

iii. Minicomputer

A computer of medium power, more than a microcomputer but less than a mainframe.

iv. Microcomputer

A microcomputer is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU). It includes a microprocessor, memory and minimal input/output (I/O) circuitry mounted on a single printed circuit board (PCB). Desktop computers, laptops, personal digital assistant (PDA), tablets & smartphones are all types of microcomputers.

The micro-computers are widely used & the fastest growing computers. These computers are the cheapest among the other three types of computers. The Micro-computers are specially designed for general usage like entertainment, education and work purposes. Well known manufacturers of Micro-computer are Dell, Apple, Samsung, Sony & Toshiba.

3. Classification of computers by Degree of Versatility

There are five types of Computers:

i. General Purpose Computer

General-purpose computers are designed to solve a large variety of problems. That is they can be given different programs to solve different types of problems. General-purpose computers can process business data as readily as they process complex mathematical formulas. General-purpose computers can store a large amount of data and the programs necessary to process them. Because general-purpose computers are so versatile, most businesses today use them. Most digital computers are general computers

and it is mainly such computers that are used in business and commercial data processing.

A PC or a Mac and other types of computers can do a huge amount of things. They can be used by different people for completely different kinds of jobs.

ii. Special Purpose Computer

Special purpose computers are designed to solve specific problems; the computer program for solving the problem is built right into the computer.

Special purpose computers have many features of general-purpose computers but are designed to handle specific problems and are not applied to other computerized activities.

For example, special purpose computers may be designed to process only numeric data or to completely control automated manufacturing processes.

Most analog computers are special-purpose computers. Special purpose computers are designed to do specific kinds of jobs. A TV, a washing machine, an iPod etc, are forms of computers, but they have only a small range of things that they can do, and are designed specifically to do them. Special purpose computers are often used as training simulators.

A simulator is a computer-controlled device for training people under simulated, or artificially created, conditions. The computer creates test conditions the trainee must respond to, it then records and evaluates the responses, providing these results to both trainee and supervisor.

iii. Machine Inbuilt computer

A computer is a machine that can be instructed to carry out sequences of arithmetic or logical ... Experimental equipment that he built in 1934 went into operation five years later, converting a portion of the telephone exchange network.

iv. High Intelligence Computer

In computer science, artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and animals. Leading AI textbooks define the field as

the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".

v. Artificial Intelligence Computer

The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

Generation of Computers

There are five generations of computers.

- First Generation
- Second Generation
- Third Generation
- Fourth Generation
- Fifth Generation

First Generation Computer (1946-1959)

The computers produced during the period 1946-1959 with the them technology are regarded as the first generation computer. These computers were manufactured with the **vacuum tubes**, electronic valves, triodes, and diodes etc. as their basic elements. These tubes were used in the arithmetic and logical operations.

- **Advantages:**

They were capable of making arithmetic and logical operations. They used the electronic valves in place of the key punch machines or the unit records machines.

- **Disadvantages:**

They were too big in size very slow, low level of accuracy and reliability. They consumed lot of electricity, generated a lot of heat and break down frequently.

Second Generation Computer

A **transistor** computer, now often called a second generation computer, is a computer which uses discrete transistors instead of vacuum tubes. The first generation of electronic computers used vacuum tubes, which generated large amounts of heat, were bulky and unreliable.

Third Generation Computer (1965-1971)

The computers of third generation used **Integrated Circuits** (ICs) in place of transistors. A single IC has many transistors, resistors, and capacitors along with the associated circuitry. ... This development made computers smaller in size, reliable, and efficient.

Fourth Generation computers (1971 – 1980)

This computer used the VLSI technology or the Very Large Scale Integrated (VLSI) circuits technology.

Therefore they were also known as the **microprocessors**. Intel was the first company to develop a microprocessor

Fifth Generation Computer

Fifth generation, VLSI technology became ULSI (Ultra Large Scale Integration) technology, resulting in the production of microprocessor chips having ten million electronic components. This generation is based on parallel processing hardware and **AI (Artificial Intelligence)** software.

Anatomy of Computer / Components of a computer

A computer consists of the following two major components:

There are two components:

i) Hardware and ii) Software

Computer			
Hardware			Software
Input	CPU	Output	Programme Languages
Keyboard Mouse Joystick,etc	ALU Control Memory	Monitor Printer Speaker, etc	Basic, Visual JAVA, C++ Fortan, Paithan,etc

i) Hardware

Computer hardware is the collection of physical parts of a computer system. This includes the computer case, monitor, keyboard, and mouse. It also includes all the parts inside the computer case, such as the hard disk drive, motherboard, video card, and many others.

Computer Anatomy

All digital computers are made up of **four** basic units. These four are :

- i) Control Unit
- ii) Arithmetic Logic Unit
- iii) Memory and
- iv) Input-Output Units

These basic units are rearranged into **three** sub-systems:

- i) Input Device
- ii) Central Processing Unit and
- iii) Output Device

Input Device and Output Device are otherwise called peripherals.

Computer is made up of the two parts: CPU and peripherals.

Input Device:

In computing, an input device is a piece of computer hardware equipment used to provide data and control signals to an information processing system such as a computer or information appliance. Examples of input devices include keyboards, mouse, scanners, digital cameras, joysticks, and microphones.

Keyboard:

The Main Typewriter like Keyboard Portion:

- i) Number keys
- ii) Special Keys and
- iii) Function keys

Cursor control key portion

Cursor-cum –numeric keyboard portion

Other keys

Mouse

Light pen

Joystick

Tracker ball

Paddle

Graphic pad and tablet

Speech synthesizer

Central Processing Unit (CPU)

The central processing unit (**CPU**) of a **computer** is a piece of hardware that carries out the instructions of a **computer** program. It performs the basic arithmetical, logical, and input/output operations of a **computer** system

Central Processing Unit

This is the heart of the computer system. That is, all operations are carried out in this unit only. It divided into three sub –units:

- i) Control Units
- ii) Arithmetic Logic Units and
- iii) Memory

i).Control Units

The control unit instructs the computer how to carry out a program's instructions. It directs the flow of data between Memory and Arithmetic Logical Units. It controls and coordinates the entire computer system.

ii). Arithmetic Logic Units

Arithmetic Logic Units performs all the arithmetic and logical operations. Arithmetic operations like:

Addition

Subtraction

Multiplication and

Logical operations such as comparison are performed in ALU.

iii). Memory

Memory is the part of the computer which holds data for processing and other information. It is also called Main Memory. The information given from the input unit is immediately stored in Main memory. It is a temporary storage (volatile memory), i.e. when the power is off, the stored information in Main Memory will be lost. To avoid this situation, we can store immediately in secondary Memory.

Secondary Storage

It is the permanent storage. The data stored in this storage will not be erased when the power is lost. It is a no... volatile memory.

Floppy disk,
Hard disk,
magnetic tapes / drums etc are some of the secondary storage devices.

Output Device

Output is the method through which information are brought out from the computer. It converts electric pulses to a form that people can understand. An output device is any piece of computer hardware equipment which converts information into human-readable form. It can be text, graphics, tactile, audio, and video. Some of the output devices are **Visual Display Units (VDU)** i.e. a **Monitor**, Printer, Graphic Output devices, Plotters, Speakers etc

Programming Languages:

Programming Languages can be broadly classified into three:

- i) Machine Languages,
- ii) Assembly languages and
- iii) High – level languages

i) Machine Languages:

The computer can execute a program written using binary digits only. This type of program is called Machine Language programs. Since these programs use only '0's and '1's it will be very difficult for developing programs for complex problems solving. Also it will be very difficult for a person to understand a machine language program written by another person. At present, computer users do not write programs using machine language. Also these programs written for execution in one computer cannot be used on another type of computer i.e., the programs were machine dependent.

ii) Assembly languages:

In Assembly languages, mnemonic codes are used to develop programs for problem solving. The program given below shows an Assembly language program to add two numbers A & B.

Assembly language is defined mainly to replace each machine code with an understandable mnemonic code. To execute an Assembly language program it should first be translated into an equivalent machine language program. Writing and understanding programs in Assembly language is easier than that of machine language. The programs written in Assembly languages were also machine dependent.

iii) **High – level languages:**

High – level languages are developed to allow application programs, which are machine independent. High – level language permits the user to use understandable codes using the language structure. In order to execute a High – level language program, it should be translated into a machine language either using a compiler or interpreter. The High – level languages commonly used are FORTRAN (FORmula TRANslation), BASIC (Beginner's All Purpose Symbolic Instruction Code), COBAL (Common Business Oriented Languages). Recently developed programming languages such as Visual Foxpro, Visual Basic (VB), Visual C++ (VC++) are more popular among the software developers. The following program written in Basic adds two given numbers.

Databases

Data is the name given to basic facts and entities such as names and numbers. Good examples of data are dates, weight, prices, costs, number of items sold, employee names, product names, addresses, tax codes, registration marks etc.

This term is used to describe basic facts about the activities of a system, may be a business house, production centre or educational institution.

Data is generally in the form of names and numbers, times, dates, weight, prices, costs, employee's name, product's name, names of books, schools, students, teachers, roll, numbers, etc.

Information

Information is a data which has been converted into a more useful or intelligible form. It is the set of data which has been converted or organised into more useful or intelligible form for direct utilization of mankind, as information helps human beings in their decision making process.

Types of data

The data input may be in the form of

- 1) Typed data or instructions in computer's language. Linguistic Raw data can be:
 - a) Numeric
 - b) Alphabetic
 - c) Alphanumeric

Then this is coded into data acceptable to computer in its language.

- 2) It may be audible, visual or audiovisual analogic data. In this case, an facing device is required to convert it into signals which are acceptable to computers.

Broadly speaking Input (Data) or Output (Information) can in the following forms: textual, graphic and pictorial, audio, visual and mechanical gestures.

All data whether linguistic or analogic is first converted into binary signals before it is accepted by computer and this process is known as digitizing.

Data Processing

By data processing we exactly mean:

- 1) Making arithmetic calculations like addition, subtraction, multiplication, division and exponentiation etc.
- 2) Making logical decision like comparing two values to find out which one is greater.

- 3) Manipulating alphabetic or alphanumeric data like word processing, letter writing, sorting in alphabetic or alphanumeric orders, editing, making catalogues etc.

Modern data processing

Modern data processing with machines falls into two broad categories:

- 1) Non-electronic data processing and
- 2) Electronic data processing

Non-electronic Data Processing (MDP)

1) Non-electronic data processing includes three types of systems:

- a) Manual systems
 - b) Semi-automatic system like punched card systems and Machines without punched cards.
-
- a) In Manual data processing (MDP), a desk calculator may be used for arithmetic calculations; a credit – card machine may be used for recording a sale; names may be alphabetized by inspection.
 - b) Punched card data processing system consists of various electro-mechanical devices, such as sorters, collators, reproducers, calculators and tabulators, which operates, whereas the electronic processing systems perform the different operations automatically.

Electronic Data Processing (EDP)

Electronic data processing (EDP) system consists of various input and output devices connected to an electronic computer.

One major difference between the two categories is that the punched –systems usually require manual intervention between the different data processing operations, whereas the electronic processing systems perform the different operations automatically.

Types of processing Methods:

Processing methods may be categorized as:

- a) Business data processing
- b) Scientific computation and Process control

Purpose of file

A file holds which is required for providing information. Some files are processed at regular interval to provide this information. (e.g. payroll file) and others will hold data which is required at irregular intervals (e.g. file containing prices of items).

Elements of a computer file:

A physical file consists of a number of records. Each record is made up of a number of fields and each field consists of a number of characters.

Record

A record is made up of a number of fields e.g., a student record, or an employee payroll record.

A record is a collection of data or information about someone or something of interest.

Data items are individual units of information, e.g. names, addresses, number of items in inventory, weekly sales total, grade in a course. A record normally consists of a set of related data items, e.g. an employee record would consist of various data items concerning a given employee.

Computational Linguistics

Computational linguistics (CL) may be thought of as the study of natural language in the intersection of linguistics and computer science. It is a relatively young scientific field that developed out of the integration of theoretical linguistics, mathematical linguistics, artificial intelligence, and software engineering.

What is Natural Language Processing or Computational Linguistics?

The idea of giving computers the ability to process human language is as old as the idea of computers themselves. The implementation and implications of that exciting idea. We introduce a vibrant interdisciplinary field with many names corresponding to its many facets, names like speech and language processing, human language technology, natural language processing, computational linguistics, and speech recognition and synthesis. The goal of this new field is to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech. One example of a useful such task is a conversational agent. The HAL 9000 computer in Stanley Kubrick's film 2001: A Space Odyssey is one of the most recognizable characters in 20th century cinema. HAL is an artificial agent capable of such advanced language behavior as speaking and understanding English, and at a crucial moment in the plot, even reading lips. It is now clear that HAL's creator, Arthur C. Clarke, was a little optimistic in predicting when an artificial agent such as HAL would be available. But just how far off was he? What would it take to create at least the language-related parts of HAL? We call programs like HAL that converse with humans in natural language conversational agents or dialogue systems. In this text we study the various components that make up modern conversational agents, including language input (automatic speech recognition and natural language understanding) and language output (dialogue and response planning and speech synthesis). Let's turn to another useful language-related task, that of making available to non-English-speaking readers the vast amount of scientific information on the Web in English. Or translating for English speakers the hundreds of millions of Web pages written in other languages like Chinese. The goal of machine translation is to automatically translate a document from one language to another. We introduce the algorithms and mathematical

tools needed to understand how modern machine translation works. Machine translation is far from a solved problem; we cover the algorithms currently used in the field, as well as important component tasks. Many other language processing tasks are also related to the Web. Another such task is Web-based question answering. This is a generalization of simple Web search, Question answering where instead of just typing keywords, a user might ask complete questions, ranging from easy to hard, like the following: • What does “divergent” mean? • What year was Abraham Lincoln born? • How many states were in the United States that year? Introduction • How much Chinese silk was exported to England by the end of the 18th century? • What do scientists think about the ethics of human cloning? Some of these, such as definition questions, or simple factoid questions like dates and locations, can already be answered by search engines. But answering more complicated questions might require extracting information that is embedded in other text on a Web page, doing inference (drawing conclusions based on known facts), or synthesizing and summarizing information from multiple sources or Web pages. In this text we study the various components that make up modern understanding systems of this kind, including information extraction, word sense disambiguation, and so on. Although the subfields and problems we’ve described above are all very far from completely solved, these are all very active research areas and many technologies are already available commercially. In the rest of this chapter, we briefly summarize the kinds of knowledge that are necessary for these tasks (and others like spelling correction, grammar checking, and so on), as well as the mathematical models that are introduced throughout the book.

The study of human languages and how they can be represented computationally and analyzed and generated algorithmically –

The cat is on the mat. --> on (mat, cat) – on (mat, cat) -->

The cat is on the mat •

In other words, building computational models of natural language comprehension and production Different Perspectives •

Cognitive Science Perspective: What happens in the brain when interpreting or producing language •

Linguistic Perspective: Similar to the cognitive perspective, but focused on the study of linguistic structures only.

- In other words, it posits various structures and tries to explain if linguistic phenomena can be explained by those structures, across languages.
- Engineering Perspective: How to build computer systems that use human language
- Cognitive Science Perspective Goal: gain an understanding of how people comprehend and produce language.
- Goal: a model that explains actual human behavior Solution must: explain psycholinguistic data be verified by experimentation

Theoretical Linguistics Perspective. In principle, coincides with the Cognitive Science Perspective. Computational linguistics can potentially help test the empirical adequacy of theoretical models.

- Linguistics is typically descriptive.
- Building computational models of the theories allows them to be empirically tested. E.g., does your grammar correctly parse all the grammatical examples in a given test suite, while rejecting all the ungrammatical examples?
- Engineering Perspective Use computational linguistics as part of a larger application:
 - Spoken dialogue systems for telephone based information systems
 - Components of Web search engines or document retrieval services

Phonology

Phonology: The study of the sound patterns of languages. We will extend this to include the letter patterns of languages.

Syntax Information Retrieval Morphology catch + PAST Spelling caught Phonemic representation K AO1 T Sound

Why study phonology in this course? Text to speech (TTS) applications include a component which converts spelled words to sequences of phonemes (= sound representations). E.g., sightPS AY1 T John P J AA1 N

Keep separate:

- Spelling (= “orthography”)
- Detailed description of pronunciation
- Abstract description of pronunciation called “phonemic representation”

Agenda: • Phonology: set of phonemes; their realizations as phones; • The phonemes are reasonably constant across a language. • The phones vary a lot within a speaker and across speakers. • Some of that variation is extremely rule-governed and must be understood: example, English “flap” (in butter).

In addition to the phonemes: syllable structure, and • Prosody. Today: stress levels: 0,1,2 • Text’s discussion of spelling errors, as a lead-in to Viterbi-ing the Minimum Edit Distance • Letter to sound (LTS)

All speakers have a set of several dozen basic pronunciation units (“phonemes”) to which they do not add (or from which delete) during their adult lifetimes. 39 phonemes in American English. • This phonemic inventory is not completely fixed and stable across the United States, but it is much more fixed and stable than is the pronunciation of these phonemes.

Articulatory phonetics

Articulatory phonetics looks at the production of sounds in the vocal tract during speech in order to describe and characterize sounds.

The field of articulatory phonetics is a subfield of phonetics that studies articulation and ways that humans produce speech. Articulatory phoneticians explain how humans produce speech sounds via the interaction of different physiological structures.

Generally, articulatory phonetics is concerned with the transformation of aerodynamic energy into acoustic energy. Aerodynamic energy refers to the airflow through the vocal tract. Its potential form is air pressure; its kinetic form is the actual dynamic airflow. Acoustic energy is variation in the air pressure that can be represented as sound waves, which are then perceived by the human auditory system as sound. Sound is produced simply by expelling air from the lungs. However, to vary the sound quality in a way useful for speaking, two speech organs normally move towards each other to contact each other to create an obstruction that shapes the air in a particular fashion. The point of maximum obstruction is called the place of articulation, and the way the obstruction forms and releases is the manner of articulation. For example, when

making a *p* sound, the lips come together tightly, blocking the air momentarily and causing a buildup of air pressure. The lips then release suddenly, causing a burst of sound. The place of articulation of this sound is therefore called bilabial, and the manner is called stop (also known as a plosive)

The production of speech involves 3 processes:

Initiation: Setting air in motion through the vocal tract.

Phonation: The modification of airflow as it passes through the larynx (related to voicing).

Articulation: The shaping of airflow to generate particular sound types (related to manner)

Articulatory phonetics refers to the “aspects of phonetics which looks at how the sounds of speech are made with the organs of the vocal tract” Ogden (2009:173). Articulatory phonetics can be seen as divided up into three areas to describe consonants. These are voice, place and manner respectively. Each of these will now be discussed separately, although all three areas combine together in the production of speech.

Voice

In English we have both **voiced** and **voiceless** sounds. A sound fits into one of these categories according to how the vocal folds behave when a speech sound is produced.

i. Voiced

ii. Voiceless

- i. Voiced:** Voiced sounds are sounds that involve vocal fold vibrations when they are produced. Examples of voiced sounds are /b,d,v,m/.

If you place two fingers on either side of the front of your neck, just below your jawbone, and produce a sound, you should be able to feel a vibrating sensation. This tells you that a sound is voiced.

- ii. **Voiceless:** Voiceless sounds are sounds that are produced with no vocal fold vibration. Examples of voiceless sounds in English are /s,t,p,f/.

Place

The vocal tract is made up of different sections, which play a pivotal role in the production of speech. These sections are called **articulators** and are what make speech sounds possible. They can be divided into two types.

The **active articulator** is the articulator that moves towards another articulator in the production of a speech sound. This articulator moves towards another articulator to form a closure of some type in the vocal tract (i.e open approximation, close, etc – define)

The **passive articulator** is the articulator that remains stationary in the production of a speech sound. Often, this is the destination that the active articulator moves towards (i.e the hard palate).

I will now talk about the different places of articulation in the vocal tract

Bilabial: Bilabial sounds involve the upper and lower lips. In the production of a bilabial sound, the lips come into contact with each other to form an effective constriction. In English, /p,b,m/ are bilabial sounds.

Labiodental: Labiodental sounds involve the lower lip (labial) and upper teeth (dental) coming into contact with each other to form an effective constriction in the vocal tract. Examples of labiodental sounds in English are /f,v/. Labiodental sounds can be divided into two types.

a) **Endolabial:** sounds produced where the upper teeth are pressed against the inside of the lower lip.

b) **Exolabial:** sounds produced where the upper teeth are pressed against the outer side of the lower lip.

- **Dental:** Dental sounds involve the tongue tip (active articulator) making contact with the upper teeth to form a constriction. Examples of Dental sounds in English are /θ, ð/. If a sound is produced where the tongue is between the upper and lower teeth, it is attributed the term 'interdental'.
- **Alveolar:** First of all, before I explain what an alveolar sound is, it's useful to locate the alveolar ridge itself. If you place your tongue just behind your teeth and move it around, you'll feel a bony sort of ridge. This is known as the alveolar ridge. Alveolar sounds involve the front portion of the tongue making contact with the alveolar ridge to form an effective constriction in the vocal tract. Examples of alveolar sounds in English are /t,d,n,l,s/.
- **Post alveolar:** Postalveolar sounds are made a little further back ('post') from the alveolar ridge. A postalveolar sound is produced when the blade of the tongue comes into contact with the post-alveolar region of your mouth. Examples of post-alveolar sounds in English are /ʃ, ʒ /.
- **Palatal:** Palatal sounds are made with the tongue body (the big, fleshy part of your tongue). The tongue body raises up towards the hard-palate in your mouth (the dome shaped roof of your mouth) to form an effective constriction. An example of a palatal sounds in English is /j/, usually spelt as <y>.
- **Velar:** Velar sounds are made when the back of the tongue (tongue dorsum) raises towards the soft palate, which is located at the back of the roof of the mouth. This soft palate is known as the velum. An effective constriction is then formed when these two articulators come into contact with each other. Examples of velar sounds in English are /k,g ŋ /.

3) Manner

In simple terms, the manner of articulation refers to the way a sound is made, as opposed to where it's made. Sounds differ in the way they are produced. When the articulators are brought towards each other, the flow of air differs according to the specific sound type. For instance, the airflow can be completely blocked off or made turbulent.

1) Stop articulations:

Stop articulations are sounds that involve a complete closure in the vocal tract. The closure is formed when two articulators come together to prevent air escaping between them. Stop articulations can be categorized according to the kind of airflow involved. The type of airflow can be oral (plosives) or nasal (nasals). I will now talk about both plosives and nasals separately.

1a) **Plosives:** are sounds that are made with a complete closure in the oral (vocal) tract. The velum is raised during a plosive sound, which prevents air from escaping via the nasal cavity. English plosives are the sounds /p,b,t,d,k,g/. Plosives can be held for quite a long time and are thus also called 'maintainable stops'.

1b) **Nasals** are similar to plosives in regards to being sounds that are made with a complete closure in the oral (vocal) tract. However, the velum is lowered during nasal sounds, which allows airflow to escape through the nasal cavity. There are 3 nasal sounds that occur in English /m,n, ŋ/

2) Fricatives:

Fricative sounds are produced by narrowing the distance between the active and passive articulators causing them to be in close approximation. This causes the airflow to become turbulent when it passes between the two articulators involved in producing a fricative sound. English fricatives are sounds such as /f,v, θ,ð, s,z, ʃ,ʒ /

3) Approximants:

Approximant sounds are created by narrowing the distance between the two articulators. Although, unlike fricatives, the distance isn't wide enough to create turbulent airflow. English has 4 approximant sounds which are /w,j,r,l/.

Vowels

When it comes to vowels, we use a different specification to describe them. We look at the vertical position of the tongue, the horizontal position of the tongue and lip position. Vowels are made with a free passage of airflow down the mid-line of the vocal tract. They are usually voiced and are produced without friction.

1) Vertical tongue position (close-open): vertical tongue position refers to how close the tongue is to the roof of the mouth in the production of a vowel. If the tongue is close, it is given the label **close**. However, if the tongue is low in the mouth when a vowel is produced, it's given the label **open**. + **close-mid/open mid (see below)**.

Some examples of open vowels: ɪ, ʊ

Some examples of close vowels: æ, ɒ,

2) Horizontal tongue position (front, mid, back): Horizontal tongue refers to where the tongue is positioned in the vocal tract in terms of 'at the front' or 'at the back' when a vowel is produced. If the tongue is at the front of the mouth it's given the label **front**, if the tongue is in the middle of the mouth it's given the label **mid** and if the tongue is at the back of the mouth it's given the label **back**.

Some examples of front vowels: ɪ, e, æ

Some examples of mid vowels: ə

Some examples of back vowels: ʌ, ɒ

3) Lip position: As is inferred, lip position concerns the position of the lips when a vowel is produced. The lips can either be **round**, **spread** or **neutral**.

Examples of round vowels: u, o

Examples of spread vowels: ɪ, ε

There are also different categories of vowels, for example: **monophthongs** and **diphthongs**.

Monophthongs: Monophthongs are vowels that are produced by a relatively stable tongue position.

Monophthongs can be divided into two categories according to their duration. These are long and short vowels and their duration is mirrored in their names.

Examples of short vowels: e, æ, ɪ, ʊ

Examples of long vowels: ɔ:, ɜ:, i:, u:

Diphthongs: Diphthongs are vowels where the tongue moves from one part of the mouth to another. They can be seen as starting of as one vowel and ending as a different vowel.

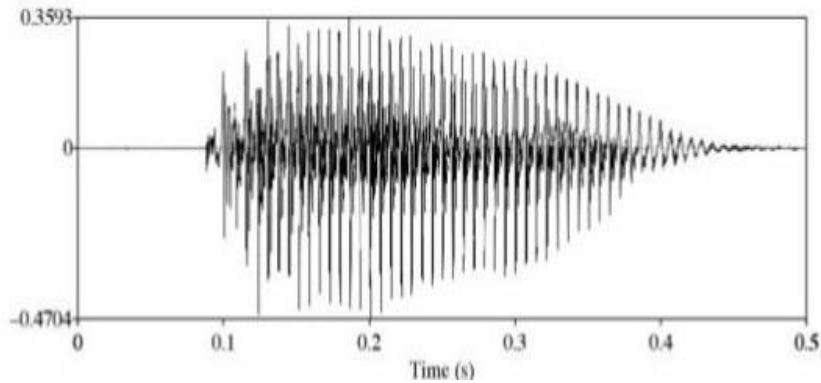
Here are some examples: /aʊ, ɪə, ɔɪ, əʊ/

Acoustic phonetics

Acoustic phonetics looks at the properties of sounds in transmission.

Acoustic phonetics is the study of the **physical properties of speech**, and aims to analyse sound wave signals that occur within speech through varying frequencies, amplitudes and durations.

One way we can analyse the acoustic properties of speech sounds is through looking at a **waveform**. Pressure changes can be plotted on a waveform, which highlights the air particles being compressed and rarefied, creating sound waves that spread outwards. A tuning fork being struck can provide an example of the pressure fluctuations in the air and how the air particles oscillate (move in one direction rhythmically) when we perceive sound.



A waveform of a vowel

– Ogden 2009: 30

Some acoustic analysis...

Frequency vs Amplitude

The **frequency** (pitch) and **amplitude** (‘loudness’ or intensity) of a sound can be analysed on a waveform. Frequency can be calculated through the number of cycles on a periodic waveform with a repeating pattern. The higher the number of cycles per second, the higher the frequency and perceived pitch. Frequency is usually expressed in **Hertz (Hz)**.

Analysing the amplitude of a waveform tells us how intense or ‘loud’ a sound is, and how much the air particles deviate. It is conventionally expressed in **decibels (dB)**.

The x-axis on a waveform corresponds to the time frame in which the sound was produced (usually in seconds or milliseconds), and the y-axis represents the amplitude.

Sine Waves vs Complex Waves

Sine waves are waveforms that have very simple, regular repeating patterns. The number of ‘cycles’ in the waveform (the number of complete repetitions in the period waveform) reflects the number of times the vocal folds have opened within the time frame displayed.

This is known as the **fundamental frequency (f₀)**, which is measured in Hertz (Hz). A frequency of 200Hz means that there are 200 hundred complete cycles per second within the waveform, so 200 times the vocal folds have opened. In reality, most speech sound waves have a rather complex pattern, and are known as **complex waves**. These are made up of two or more simple sine waves, and the fundamental frequency can also be

calculated on complex waveforms by counting the number of cycles per second on a waveform.

Periodic vs Aperiodic sound waves

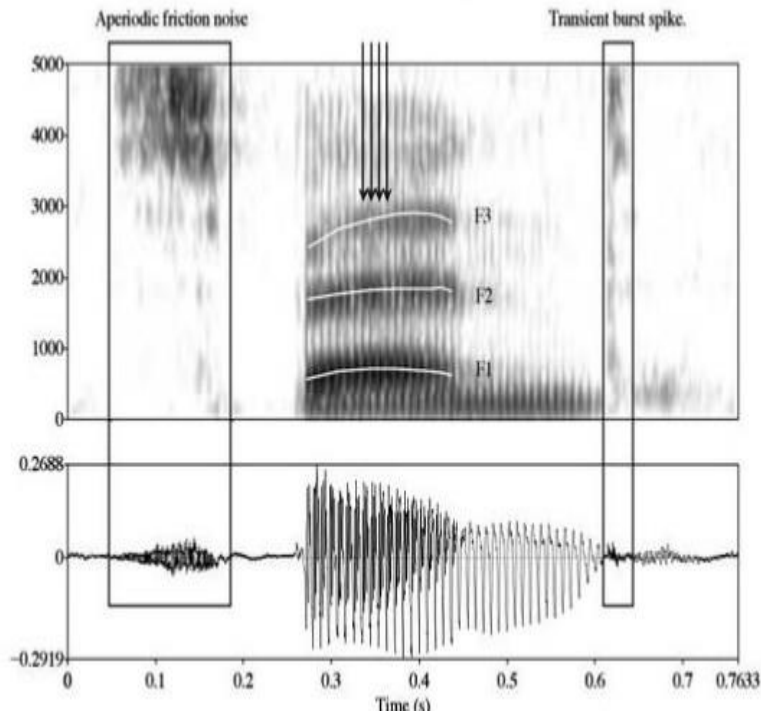
Sine and complex waveforms are **periodic**, meaning their cycles are regular and repetitive. The types of speech sounds that would appear as a periodic sound wave are voiced sounds, such as vowels or nasals. Since such sounds have regularly repeating waveforms, they can also be decoded through '**Fourier analysis**' which breaks down the component sine waves. This type of graph is called a **spectrum**, which does not measure time. Instead, the x-axis measures frequency, and the y-axis represents the sound pressure level.

The fundamental frequency on this type of graph can be worked out by selecting the lowest frequency component of this complex wave. This is usually the first complete peak on the spectrum. From this fundamental frequency peak, **harmonics** occur at evenly spaced integer multiples. Harmonics are known as the '**natural resonances**' within the vocal tract, which are the amplified frequencies. On the spectrum, these correspond to each peak.

On the other hand, speech sounds can also be **aperiodic** when analysing them acoustically. This means that they do not have a regular repeating pattern, rather, they have a very random pattern meaning that a fundamental frequency cannot be calculated. This means that the aperiodic speech sounds are voiceless, such as a voiceless fricative.

Spectrograms

Another way to analyse a sound acoustically is through looking at a **spectrogram**. They provide much more complex information than what we can see on a waveform. Similarly to waveforms, time is displayed on the x-axis, but the y-axis measures the frequency of the sound. Amplitude is represented by the darkness in the acoustic energy. The louder the sound, the darker it appears on a spectrogram and is therefore more intense. Spectrograms allow us to see the high frequency energy that comes with aperiodic sounds.



A spectrogram of the word . The periodic, aperiodic and transient sounds are marked.
 Ogden 2009: 32

Transients vs Continuous sounds

Transients are a form of aperiodic sound. It is a sound that builds up pressure behind a closure, and then has a sudden burst/release which shows up as a spike on a waveform. On a spectrogram, the closure shows up as a blank space before a dark vertical band of acoustic energy to represent the release. A typical transient sound would be a plosive, such as [p] or [b] in the English sound system.

Continuous sounds are another form of aperiodicity. Unlike a voiced vowel sound, this would show up as an irregular, random pattern on the waveform. On the spectrogram, this would be represented by high frequency acoustic energy which is dark and intense, and therefore has high amplitude. This is typical of many voiceless fricatives, such as [f] and [s] in the English sound system.

Voicing on a spectrogram

As established earlier, voiced sounds are periodic signals. A fundamental frequency can be calculated due to the regular openings of the vocal folds as they vibrate. On a waveform, this would be highlighted by a periodic sound wave. On a spectrogram, there are two specific visual elements to look out for, which resemble a voiced sound. The first is the **vertical striations** (they look like vertical wavy lines on the spectrogram), which correspond to this opening of the vocal folds, and when air flows through them every time. The other visual clue is the dark horizontal bands which are typical of vowels, approximants and nasals. These are called **formants**, which are the natural resonances of the vocal tract (earlier, they were described as harmonics). The size and shape of the vocal tract can be modified to allow these formants to vary. This can be done by changing the tongue position, lip position, etc.

Prosodic features

In Linguistics, **prosody** is concerned with those elements of speech that are not individual phonetic segments (vowels and consonants) but are properties of syllables and larger units of speech, including linguistic functions such as intonation, tone, stress, and rhythm. Such elements are known as **suprasegmentals**.

Prosody may reflect various features of the speaker or the utterance: the emotional state of the speaker; the form of the utterance (statement, question, or command); the presence of irony or sarcasm; emphasis, contrast, and focus. It may otherwise reflect other elements of language that may not be encoded by grammar or by choice of vocabulary.

Suprasegmental phenomena in synthetic speech should reflect the linguistic structure of the input text. An algorithm is described, which establishes the prosodic sentence structure (PSS). This can be achieved without exhaustive syntactic parsing, using a dictionary of 550 function words. Subsequently, phrase and accent locations are derived from the PSS; accentuation is also affected by some semantic and contextual information. Comparison of the resulting sentence prosody with that of a human (professional) speaker shows that more detailed syntactic analysis may be necessary. Most of the accentuation errors are caused by semantic, pragmatic and contextual factors.

These factors can only partly be imitated (using heuristics), since the relations between linguistic representations and real-world knowledge are not yet fully understood.

Prosodic features are **features** that appear when we put sounds together in connected speech. It is as important to teach learners **prosodic features** as successful communication depends as much on intonation, stress and rhythm as on the correct pronunciation of sounds. Intonation, stress and rhythm are **prosodic features**.

Computational Phonology rules

Computational phonology means to understand modern speech recognition and speech synthesis technology.

Text-to –speech synthesis is to take a sequence of text words and produce as output an acoustic waveform.

The uses of speech recognition and synthesis are manifold including automatic dictation transcription; Speech based interferences to computer and telephones, voice based Input and Output for the disabled and many others. The speech recognition and Text – to – Speech systems.

How words are pronounced in terms of individual speech units are called phones. A speech recognition system needs to have a pronunciation for every word it can recognize, and a text – to – speech system need to have a pronunciation for every word it can say.

Speech synthesis is the artificial production of human speech.

A computer system used for this purpose is called a **speech computer** or **speech synthesizer**, and can be implemented in software or hardware products.

- **Text-to-Speech (TTS)**
 - **Speech –to - Text**
-
- **Text-to-Speech (TTS)**

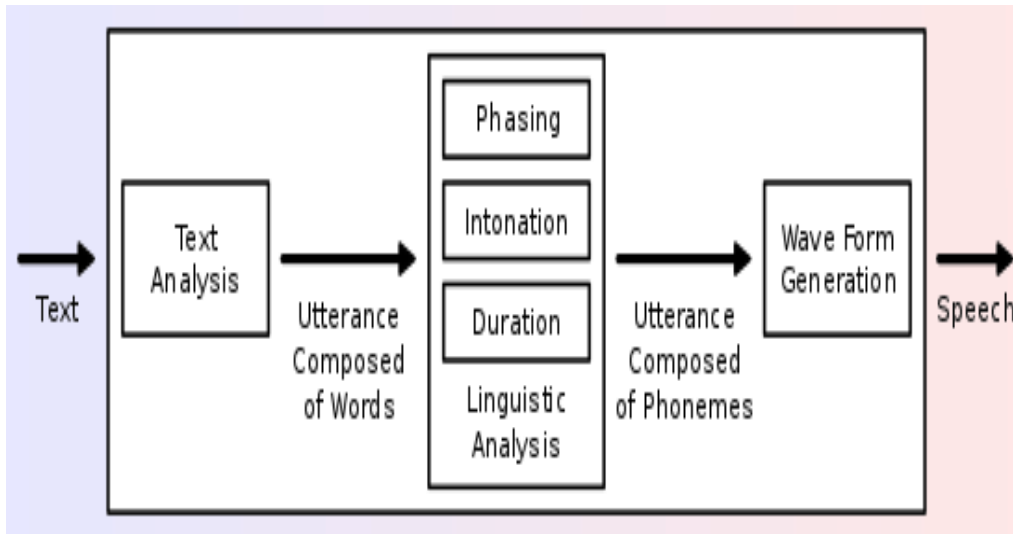
A **text-to-speech (TTS)** system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity.

For specific usage domains, the storage of entire words or sentences allows for high-quality output.

Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output. The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood clearly. An intelligible text-to-speech program allows people with Visual impairments or reading to listen to written words on a home computer.

- Many computer operating systems have included speech synthesizers since the early 1990s.



- Speech-to-Text documentation
- Speech-to-Text enables easy integration of Google speech recognition technologies into developer applications.
- Send audio and receive a text transcription from the Speech-to-Text API service.
- Powerful speech recognition
- Google Speech-to-Text enables developers to convert audio to text by applying powerful neural network models in an easy-to-use API.
- The API recognizes more than 120 languages and variants to support your global user base.
- You can enable voice command-and-control, transcribe audio from call centers, and more.
- It can process real-time streaming or prerecorded audio, using Google’s machine learning technology.
- Convert your speech to text right

Speech synthesis

Introduction The ability of human beings to interact through spoken language is considered as an intelligent trait that set humans apart from other animals. Humans use speech, an acoustic signal, to exchange ideas over long distance even when visual communication is impossible. The primary goal of spoken communication is to convey

an idea in speaker's mind; this idea or a concept or information is encoded in the speech signal. Speech signal also carries auxiliary information such as speaker characteristics, the language being used, the acoustic environment etc. Retrieval of such information from speech signal by machine is called automatic recognition. In particular, the decoding of the linguistic information from the speech signal by machine is called Automatic Speech Recognition (ASR). Extraction of information about the speaker and language by computers are called automatic speaker recognition and language identification respectively. These topics will be covered in four modules of this paper. Speech is produced by human vocal organ, and perceived by the listener's brain via processing of speech signal by human auditory system. In contrast, speech signal is processed by computer in case of ASR.

The goal of machine speech processing is to compute certain attributes, called speech features, from speech that will help in characterizing and identifying speech sounds present in time waveform. The sequence of speech sounds thus hypothesized are utilized to decode an utterance in the form of words, phrases and sentences. This step exploits linguistic knowledge such as lexicon, grammar. The processing of speech signal by computer, also called feature extraction, is described in this module. 1.2Speech Production Knowledge of production of various speech sounds will help us to understand properties of speech sounds in time as well as frequency domains. A brief overview of speech production will be given here; for detailed information, see modules on the articulatory process (Pandey 2015).

When a person intends to convey an idea through speech, she forms a sentence in a language of her choice. Then, the human speech production apparatus, shown schematically in Fig. 1, starts generating a sequence of sounds by utilizing energy in the form of air pressure in lungs to radiate speech waveform through mouth (and sometimes through nose and throat walls). For illustration, let us consider the production of sound "a", denoted by the first letter in Indian language alphabets. Initially, the vocal folds in the larynx are brought close so as to obstruct the passage of air to and from lungs. The consequent air pressure from below the vocal folds coerce the vocal folds to move apart; then, air escapes through the resultant opening (called glottis) to oral cavity shaded yellow in Fig. 1. Due to flow of air through glottis, the air pressure between the vocal

folds reduces. This compels them to come together, and stop the air flow. This process of starting and stopping of air flow repeats several times a second. This rate is known popularly as the 'pitch' frequency. Pitch frequency depends on the physiological characteristics of the speaker. The pitch of an adult male speaker could be as low as 75Hz and as high as 250Hz. The pitch for female speakers and children is generally higher; the typical range is [200, 500]. A schematic diagram of speech production apparatus [source: Garay, W.(2012)]”.

The train of air pockets coming through glottis causes resonant vibration of air molecules in the vocal cavity. This is similar to blowing air into a flute. Depending on the size and shape of this resonant air cavity, the energy in sound coming out of mouth gets concentrated at certain frequencies called formant frequencies. Formant frequency will be high if the dimension (for example, the length) of the vocal cavity is small, and vice versa. Such an inverse relationship between the length of resonant air cavity and the resonance frequency is easily seen in case of sound produced by a flute. For example, if all the side holes of a flute are closed, the length of the resonant tube is long, and equal to length of the flute; consequently, a low frequency tune is heard. In contrast, If all holes are kept open, there will be several resonant cavities of short length; the resultant tune will sound shrill due to more energy at the high frequency. In order to different vowels such as “I”, “u”, the position of jaw, tongue, lips are altered. The consequent change in resonant vocal cavity configuration, results in different set of values of formant frequencies. Thus, formant frequencies are signatures of vowels. Vowels are defined as speech sounds in whose production, there is no obstruction of air flow between glottis and mouth. In contrast, if there is such an obstruction, the resultant speech sounds are called consonants.

The consonants are called fricatives if the obstruction is narrow, but partial. Complete stoppage of airflow momentarily results in production of stop consonants. If the air is permitted to come out through nose but not through mouth, the corresponding consonants are called nasals. 1.3Speech Analysis in time and frequency domains time waveform of a spoken word “tiruvanthpuram”. In this graph, time is on the horizontal axis and amplitude is on the vertical axis. This word has 5 vowels and 8 consonants. Location of vowels can be identified by their high amplitude. However, it is difficult to

identify the vowels by visual inspection (without listening) because the vowels appear similar in the time domain. However, as we discussed in the previous para, different vowels have different sets formant frequencies.

These formants appear as peaks in frequency spectrum. Hence, it is easier to recognise vowels (and speech sounds, in general) in frequency domain. This is the motivation for carrying out frequency analysis of speech waveform with the goal of recognising sounds. Time waveform of utterance "Tiruvanthpuram" The time waveform shown in Fig. 2 contains several sounds. Since the properties of sounds are different, we need to do a frequency analysis of short segments of wave. The outcome of frequency analysis of a segment of vowel waveform of 0.025s duration is shown in Fig. 3. This graph is known as frequency spectrum or power spectrum or simply spectrum. Here, frequency (in Hz) varies along the horizontal axis, and energy (equivalently power) varies along the vertical axis. The spectrum displays the amount of energy present in this vowel at various frequencies. Generally, the range of energy values is large; the highest energy value could be ten thousand times the lowest energy value. So, all the values cannot be shown on a linear scale. Hence the energy values (on the vertical axis) are plotted on a logarithmic (dB) scale. 4 Figure 3. Log power spectrum of a vowel (the zagged yellow curve). This curve is the sum of (i) regular, frequently occurring narrow peaks due to pitch, and (ii) slowly varying broad peaks due to vocal tract resonance. The smoothing of the yellow curve yields the smooth purple curve from which formant peaks (signatures of speech sounds) can be easily extracted, leading to recognition of sounds (ASR). Two types of peaks are visible in the spectrum: i. narrow peaks that occur at regular frequency interval ii. broad, irregularly spaced peaks in spectral envelope (curve joining the tops of narrow peaks). The broad peaks are called formants resulting from resonance in vocal cavity. The height, width and the frequency at which these peaks occur depend on the vowel spoken.

,

Speech Production

Speech is produced by human vocal organ, and perceived by the listener's brain via processing of speech signal by human auditory system. In contrast, speech signal is processed by computer in case of ASR. The goal of machine speech processing is to

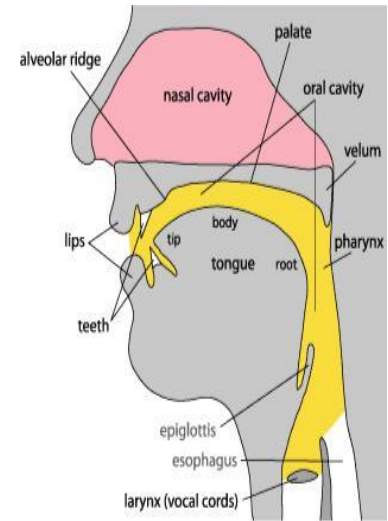
compute certain attributes, called speech features, from speech that will help in characterizing and identifying speech sounds present in time waveform. The sequence of speech sounds thus hypothesized are utilized to decode an utterance in the form of words, phrases and sentences. This step exploits linguistic knowledge such as lexicon, grammar.

The processing of speech signal by computer, also called **feature extraction**, is described in this module.

Knowledge of production of various speech sounds will help us to understand properties of speech sounds in time as well as frequency domains. A brief overview of speech production will be given here; for detailed information, see modules on the articulatory process (Pandey 2015).

When a person intends to convey an idea through speech, she forms a sentence in a language of her choice. Then, the human speech production apparatus, shown schematically in Fig. 1, starts generating a sequence of sounds by utilizing energy in the form of air pressure in lungs to radiate speech waveform through mouth (and sometimes through nose and throat walls).

For illustration, let us consider the production of sound “a”, denoted by the the first letter in Indian language alphabets. Initially, the vocal folds in the larynx are brought close so as to obstruct the passage of air to and from lungs.



The consequent air pressure from below the vocal

folds coerces the vocal folds to move apart; then, air escapes through the resultant opening (called **glottis**) to oral cavity shaded yellow in Fig. 1. Due to flow of air through glottis, the air pressure between the vocal folds reduces. This compels them to come together, and stop the air flow. This process of starting and stopping of air flow repeats several times a second. This rate is known popularly as the ‘ **pitch** ’ frequency. Pitch frequency depends on the physiological characteristics of the speaker. The pitch of an adult male speaker could be as low as 75Hz and as high as 250Hz. The pitch for female speakers and children is generally higher; the typical range is [200, 500].

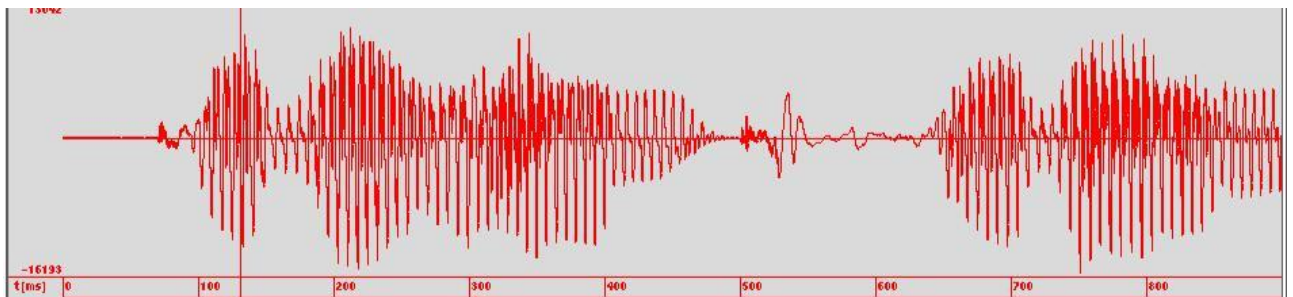
The train of air pockets coming through glottis causes resonant vibration of air molecules in the vocal cavity. This is similar to blowing air into a flute. Depending on the size and shape of this resonant air cavity, the energy in sound coming out of mouth gets concentrated at certain frequencies called **formant frequencies**. Formant frequency will be high if the dimension (for example, the length) of the vocal cavity is small, and vice versa. Such an inverse relationship between the length of resonant air cavity and the resonance frequency is easily seen in case of sound produced by a flute. For example, if all the side holes of a flute are closed, the length of the resonant tube is long, and equal to length of the flute; consequently, a low frequency tune is heard. In contrast, if all holes are kept open, there will be several resonant cavities of short length; the resultant tune will sound shrill due to more energy at the high frequency.

In order to differentiate different vowels such as “I”, “u”, the position of jaw, tongue, lips are altered. The consequent change in resonant vocal cavity configuration, results in different set of values of formant frequencies. Thus, formant frequencies are signatures of vowels. Vowels are defined as speech sounds in whose production, there is no obstruction of air flow between glottis and mouth. In contrast, if there is such an obstruction, the resultant speech sounds are called consonants. The consonants are called fricatives if the

obstruction is narrow, but partial. Complete stoppage of airflow momentarily results in production of stop consonants. If the air is permitted to come out through nose but not through mouth, the corresponding consonants are called nasals.

Speech Analysis in time and frequency domains

Fig. 2 shows time waveform of a spoken word “tiruvanthpuram”. In this graph, time is on the horizontal axis and amplitude is on the vertical axis. This word has 5 vowels and 8 consonants. Location of vowels can be identified by their high amplitude. However, it is difficult to identify the vowels by visual inspection (without listening) because the vowels appear similar in the time domain. However, as we discussed in the previous para, different vowels have different sets formant frequencies. These formants appear as peaks in frequency spectrum. Hence, it is easier to recognise vowels (and speech sounds, in general) in frequency domain. This is the motivation for carrying out **frequency analysis** of speech waveform with the goal of recognising sounds.



The time waveform shown in Fig. 2 contains several sounds. Since the properties of sounds are different, we need to do a frequency analysis of short segments of wave. The outcome of frequency analysis of a segment of vowel waveform of 0.025s duration is shown in Fig. 3. This graph is known as frequency spectrum or power spectrum or simply spectrum. Here, frequency (in Hz) varies along the horizontal axis, and energy (equivalently power) varies along the vertical axis. The spectrum displays the amount of energy present in this vowel at various frequencies. Generally, the range of energy values

is large; the highest energy value could be ten thousand times the lowest energy value. So, all the values cannot be shown on a linear scale. Hence the energy values (on the vertical axis) are plotted on a logarithmic (dB) scale.

Figure 3. Log power spectrum of a vowel (the zagged yellow curve). This curve is the sum of (i) regular, frequently occurring narrow peaks due to pitch, and (ii) slowly varying broad peaks due to vocal tract resonance. The smoothing of the yellow curve yields the smooth purple curve from which formant peaks (signatures of speech sounds) can be easily extracted, leading to recognition of sounds (ASR).

Two types of peaks are visible in the spectrum:

- i. narrow peaks that occur at regular frequency interval
- ii. broad, irregularly spaced peaks in spectral envelope (curve joining the tops of narrow peaks).

The broad peaks are called formants resulting from resonance in vocal cavity. The height, width and the frequency at which these peaks occur depend on the vowel spoken. Thus information about formants can be used to identify vowels and other sounds. The regular, narrow peaks are called pitch harmonics. These peaks represent the periodic opening and closing of vocal folds at pitch frequency. Firstly, these regular peaks hardly carry information about the identity of speech sound. More importantly, if the speaker speaks the vowel with a high pitch (e.g., when she is angry), the interval between these pitch related regular peaks in spectrum increases. This results in a different visual appearance of spectrum even though the speaker has spoken the same vowel. Therefore, it is important to remove the closely spaced pitch peaks from the graph; this enables us (or machines in ASR) to focus on formants (signatures of speech sounds).

UNIT 2

Computational Morphology and Syntax

Words and Morphemes

In traditional grammar, words are the basic units of analysis. Grammarians classify words according to their parts of speech and identify and list the forms that words can show up in. Although the matter is really very complex, for the sake of simplicity we will begin with the assumption that we are all generally able to distinguish words from other linguistic units. It will be sufficient for our initial purposes if we assume that words are the main units used for entries in dictionaries. In a later section, we will briefly describe some of their distinctive characteristics.

Words are potentially complex units, composed of even more basic units, called morphemes. A **morpheme** is the smallest part of a word that has grammatical function or meaning (NB not the smallest unit of meaning); we will designate them in braces—{ }. For example, *sawed*, *sawn*, *sawing*, and *saws* can all be analyzed into the morphemes {saw} + {- ed}, {- n}, {- ing}, and {- s}, respectively. None of these last four can be further divided into meaningful units and each occurs in many other words, such as *looked*, *mown*, *coughing*, *bakes*.

{Saw} can occur on its own as a word; it does not have to be attached to another morpheme. It is a **free morpheme**. However, none of the other morphemes listed just above is free. Each must be **affixed** (attached) to some other unit; each can only occur as a part of a word. Morphemes that must be attached as word parts are said to be **bound**.

Affixes are classified according to whether they are attached before or after the form to which they are added. **Prefixes** are attached before and **suffixes** after. The bound morphemes listed earlier are all suffixes; the {re- } of *resaw* is a prefix.

Free Morphemes and Bound Morphemes

Morphemes that can stand alone to function as words are called free morphemes. They comprise simple words (i.e. words made up of one free morpheme) and compound words (i.e. words made up of two free morphemes).

Examples:

Simple words: **the, run, on, well**

Compound words: **keyboard, greenhouse, bloodshed, smartphone**

Morphemes that can only be attached to another part of a word (cannot stand alone) are called bound morphemes.

Examples:

pre-, dis-, in-, un-, -ful, -able, -ment, -ly, -ise

pretest, discontent, intolerable, receive

Complex words are words that are made up of both free morpheme(s) and bound morpheme(s), or two or more bound morphemes.

Roll your mouse over the words below to see how many morphemes are there and whether they are free morphemes or bound morphemes.

Root, derivational, and inflectional morphemes

Besides being bound or free, morphemes can also be classified as root, derivational, or inflectional. A **root** morpheme is the basic form to which other morphemes are attached. It provides the basic meaning of the word. The morpheme {saw} is the root of *sawers*. **Derivational** morphemes are added to forms to create separate words: {-er} is a derivational suffix whose addition turns a verb into a noun, usually meaning the person or thing that performs the action denoted by the verb. For example, {paint}+{-er} creates *painter*, one of whose meanings is “someone who paints.”

Inflectional morphemes do not create separate words. They merely modify the word in which they occur in order to indicate grammatical properties such as plurality, as the {-s} of *magazines* does, or past tense, as the {ed} of *babecued* does. English has eight inflectional morphemes, which we will describe below.

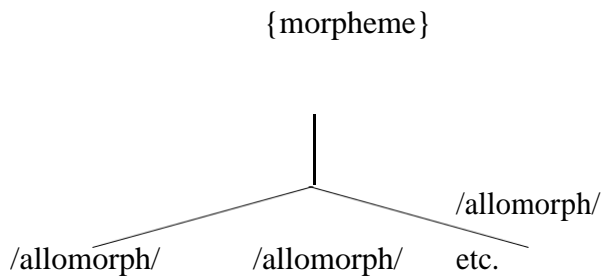
We can regard the root of a word as the morpheme left over when all the derivational and inflectional morphemes have been removed. For example, in *immovability*, {im-}, {-abil}, and {-ity} are all derivational morphemes, and when we remove them we are left with {move}, which cannot be further divided into meaningful pieces, and so must be the word's root.

We must distinguish between a word's root and the forms to which affixes are attached. In *moveable*, {-able} is attached to {move}, which we've determined is the word's root. However, {im-} is attached to *moveable*, not to {move} (there is no word *immove*), but *moveable* is not a root. Expressions to which affixes are attached are called **bases**. While roots may be bases, bases are not always roots.

Morphemes, Allomorphs and Morphs

The English plural morpheme {-s} can be expressed by three different but clearly related phonemic forms /z/ or /z/, /z/, and /s/. These three have in common not only their meaning, but also the fact that each contains an alveolar fricative phoneme, either /s/ or /z/. The three forms are in complementary distribution, because each occurs where the others cannot, and it is possible to predict just where each occurs: /z/ after sibilants (/s, z, , , t , d /), /z/ after voiced segments, and /s/ everywhere else. Given the semantic and phonological similarities between the three forms and the fact that they are in complementary distribution, it is reasonable to view them as contextual pronunciation variants of a single entity. In parallel with phonology, we will refer to the entity of which the three are variant representations as a **morpheme**, and the variant forms of a given morpheme as its **allomorphs**. When we wish to refer to a minimal grammatical form

merely as a form, we will use the term **morph**. Compare these terms and the concepts behind them with phoneme, allophone, and phone. (Hint: note the use of / /, [], and { }.)



Words

Words are notoriously difficult entities to define, both in universal and in language specific terms. Like most linguistic entities, they look in two directions—upward toward larger units of which they are parts (toward phrases), and downward toward their constituent morphemes. This, however, only helps us understand words if we already understand how they are combined into larger units or divided into smaller ones, so we will briefly discuss several other criteria that have been proposed for identifying them.

One possible criterion is spelling: in written English text, we tend to regard as a word any expression that has no spaces within it and is separated by spaces from other expressions. While this is a very useful criterion, it does sometimes lead to inconsistent and unsatisfactory results. For instance, *cannot* is spelled as one word but *might not* as two; compounds (words composed of two or more words; see below) are inconsistently divided (cf. *influx*, *in-laws*, *goose flesh*, *low income* vs. *low-income*).

Words tend to resist interruption; we cannot freely insert pieces into words as we do into sentences. For example, we cannot separate the root of a word from its inflectional ending by inserting another word, as in **sock-blue-s* for *blue socks*. Sentences, in contrast, can be interrupted. We can insert adverbials between subjects and predicates: *John quickly erased his fingerprints*. By definition, we can also insert the traditional interjections: *We will, I believe, have rain later today*.

In English, though by no means in all languages, the order of elements in words is quite fixed. English inflections, for example, are suffixes and are added after any derivational morphemes in a word. At higher levels in the language, different orders of elements can differ in meaning: compare *John kissed Mary* with *Mary kissed John*. But we do not contrast words with prefixed inflections with words with suffixed inflections. English does not contrast, for example, *piece + s* with *s + piece*.

In English, too, it is specific individual words that select for certain inflections. Thus the word *child* is pluralized by adding {- ren}, *ox* by adding {- en}. So if a form takes the {-en} plural, it must be a word.

So **words** are units composed of one or more morphemes; they are also the units of which phrases are composed.

Inflectional Morphology

Inflectional morphemes, as we noted earlier, alter the form of a word in order to indicate certain grammatical properties. English has only eight inflectional morphemes, listed in Table 1, along with the properties they indicate.

Except for {-en}, the forms we list in Table 1 are the **regular** English inflections. They are regular because they are the inflections added to the vast majority of verbs, nouns, adjectives, and adverbs to indicate grammatical properties such as tense, number, and degree.

They are also the inflections we typically add to new words coming into the language, for example, we add {-s} to the noun *throughput* to make it plural. When we borrow words from other languages, in most cases we add the regular English inflections to them rather than borrow the inflections they had in their home languages; for example, we pluralize *operetta* as *oper-ettas* rather than as *operette* as Italian does; similarly, we sing *oratorios* rather than *oratori*. [Thanks to Paula Malpezzi-Price for help with these

examples.] The regular inflections are the default inflections that learners tend to use when they don't know the correct ones (for example, *grewed* rather than *grew*).

nouns:	{-s}	Plural	(the birds)
noun			(the bird's)
phrases:	{-s}	genitive/possessive	song)
adjectives/adverbs:	{-er}	Comparative	(faster)
	{-est}	Superlative	(fastest)
verbs: {-s}		3rd person singular present tense	(proves)
	{-ed}	past tense	(proved)
	{-		
	ing}	progressive/present participle	(is proving)
		past	
	{-en}	participle	(has proven)
			(was proven)

table 1: the eight english inflectional morphemes

[Note: the regular past participle morpheme is {-ed}, identical to the

past tense form {-ed}. We use the irregular past participle form {-en} to

distinguish the two.]

However, because of its long and complex history, English (like all languages) has many **irregular** forms, which may be irregular in a variety of ways. First, irregular words may use different inflections than regular ones: for example, the modern past participle inflection of a regular verb is {-ed}, but the past participle of *freeze* is *frozen* and the past participle of *break* is *broken*. Second, irregular forms may involve internal vowel changes, as in *man/men*, *woman/women*, *grow/grew*, *ring/rang/rung*. Third, some forms derive from historically unrelated forms: *went*, the past tense of *go*, historically

was the past tense of a different verb, *wend*. This sort of realignment is known as **suppletion**. Other examples of suppletion include *good, better, and best*, and *bad, worse, and worst*. (As an exercise, you might look up *be, am, and is* in a dictionary that provides etymological information, such as the American Heritage.) Fourth, some words show no inflectional change: *sheep* is both singular and plural; *hit* is both present and past tense, as well as past participle. Fifth, many borrowed words, especially nouns, have irregular inflected forms: *alumnae* and *cherubim* are the plurals of *alumna* and *cherub*, respectively.

Irregular forms demonstrate the abstract status of morphemes. Thus the word *men* **realizes** (represents, makes real) the two morphemes {man} and {plural}; *women* realizes {woman} and {plural}; *went* realizes {go} and {past tense}. Most grammar and writing textbooks contain long lists of these exceptions.

As a final issue here we must note that different groups of English speakers use different inflected forms of words, especially of verbs. When this is the case, the standard variety of the language typically selects one and rejects the others as non-standard, or, illogically, as “not English,” or worse. For example, many English speakers use a single form of *be* in the past tense (*was*) regardless of what the subject of its clause is. So they will say, *We was there yesterday*. This is an uncontroversial issue: *was* in instances like this is universally regarded as non-standard. Other forms are more controversial. For example, what is the past tense of *dive*—*dived* or *dove*? How are *lie* and *lay* to be used? How does your dictionary deal with such usage issues?.

Derivational Morphology

Derivation is the process of creating separate but morphologically related words. Typically, but not always, it involves one or more changes in form. It can involve prefixing, as in *resaw*, and suffixing, as in *insawing, sawer, sawable*.

Another type of derivation, while not visible, is at least audible. It involves a change in the position of the primary stress in a word. Compare:

(3) permit (noun) per mit (verb)

contact (noun) con tact (verb)

perfect (adj.) per fect (verb)

convert (noun) con vert (verb)

In some derivationally related word pairs, only a feature of the final con-sonant changes, usually its voicing:

(4) advice Advise /s/ /z/

belief Believe /f/ /v/

mouth Mouthe // //

breath Breathe // //

In some cases adding a derivational morpheme induces a change in a stressed vowel:

(5) divine Divinity /a/ //

profane profanity /e/ //

serene Serenity /i/ //

In other cases, the addition of a suffix triggers a change in the final con-sonant of the root. For example, an alveolar consonant becomes palatal with the same voicing value:

(6) part partial /t/ //

face facial /s/ //

seize seizure /z/ //

remit remission /t/ //

In a multi-syllabic word with a stressed tense vowel, the palatalization may be accompanied by a laxing of that vowel:

(7) collide collision /d/ // /a / //

elide elision /d/ // /a / //

Sometimes the addition of a derivational affix requires a change in the stress pattern, with consequential changes in the pronunciations of the vowels. In most cases an unstressed vowel is pronounced as schwa:

(8) telegraph telegraphy

regal regalia

tutor tutorial

In still other cases we find suffixing, stress migration with change of vowel quality, and change of consonant:

(9) approve approbation /u/ // /v/ /b/

Additionally, English allows us to change a word's part of speech without any change of form. As a result, identical forms may belong to different parts of speech, e.g., *saw* the noun and *saw* the verb:

(5) a. This saw is too dull. (noun) b. Don't saw that board. (verb)

Other examples include *hit*, *buy*, *dust*, *autograph*, *brown-bag*, which can all be both verbs and nouns. Change of part of speech without any corresponding formal change is called **conversion** (also **functional shift** or **zero derivation**). There is more on this topic in our chapter on Major Parts of Speech.

Compounding

The italicized words in (11) are created by combining *saw* with some other word, rather than with a bound morpheme.

(9) a. A *sawmill* is a noisy place.

Every workshop should have a *chain saw*, a *table saw*, a *jig-saw*, a *hack saw*, and a *bucksaw*.

Sawdust is always a problem in a woodworker's workshop.

Sawing horses are useful and easily made.

Such words are called **compounds**. They contain two or more words (or more accurately, two or more roots, all, one, or none of which may be bound; cf. *blueberry* with two free morphemes, and *astronaut* with two bound morphemes). Generally, one of the words is the head of the compound and the other(s) its modifier(s). In *bucksaw*, *saw* is the head, which is modified by *buck*. The order is significant: compare *pack rat* with *rat pack*. Generally, the modifier comes before the head.

In ordinary English spelling, compounds are sometimes spelled as single words, as in *sawmill*, *sawdust*; sometimes the parts are connected by a hyphen, as in *jig-saw*; and sometimes they are spelled as two words, as in *chain saw*, *oil well*. (Dictionaries may differ in their spellings.) Nonetheless, we are justified in classifying all such cases as compound words regardless of their conventional spelling for a variety of reasons.

First, the stress pattern of the compound word is usually different from the stress pattern in the phrase composed of the same words in the same order. Compare:

(12) compound	phrase
White House	white house
funny farm	funny farm
blackbird	black bird
flatcar	flat car

In the compounds the main stress is on the first word; in the phrases the main stress is on the last word. While this pattern does not apply to all compounds, it is so generally true that it provides a very useful test.

Second, the meaning of the compound may differ to a greater or lesser degree from that of the corresponding phrase. A *blackbird* is a species of bird, regardless of its color; a *black bird* is a bird which is black, regardless of its species. A *trotting-horse* is a kind of horse, regardless of its current activity; a *trotting horse* must be a horse that is currently trotting. So, because the meanings of compounds are not always predictable from the meanings of their constituents, dictionaries often provide individual entries for them. They do not do this for phrases, unless the meaning of the phrase is **idiomatic** and therefore not derivable from the meanings of its parts and how they are put together, e.g., *raining cats and dogs*. Generally the meaning of a phrase is predictable from the meanings of its constituents, and so phrases need not be listed individually. (Indeed, because the number of possible phrases in a language is infinite, it is in principle impossible to list them all.)

Third, in many compounds, the order of the constituent words is different from that in the corresponding phrase:

(13) compound	phrase
sawmill	mill for sawing
sawing horse	horse for sawing
sawdust	dust from sawing

Fourth, compound nouns allow no modification to the first element. This contrasts with noun phrases, which do allow modification to the modifier: compare **a really-blackbird* and *a really black bird*.

There are a number of ways of approaching the study and classification of compound words, the most accessible of which is to classify them according to the part of speech of the compound and then sub-classify them according to the parts of speech of its constituents. Table 2 is based on discussion in Bauer (1983).

1. Compound nouns

Noun + noun: bath towel; boy-friend; death blow

Verb + noun: pickpocket; breakfast

Noun + verb: nosebleed; sunshine

Verb + verb: make-believe

Adjective + noun: deep structure; fast-food

Particle + noun: in-crowd; down-town

Adverb + noun: now generation

Verb + particle: cop-out; drop-out

Phrase compounds: son-in-law

2. Compound verbs

Noun + verb: sky-dive

Adjective + verb: fine-tune

Particle + verb: overbook

Adjective + noun: brown-bag

3. Compound adjectives

Noun + adjective: card-carrying; childproof

Verb + adjective: fail safe

Adjective + adjective: open-ended

Adverb + adjective: cross-modal

Particle + adjective: over-qualified

Noun + noun: coffee-table

Verb + noun: roll-neck

Adjective + noun: red-brick; blue-collar

Particle + noun: in-depth

Verb + verb: go-go; make-believe

Adjective/Adverb + verb: high-rise;

Verb + particle: see-through; tow-away

4. Compound adverbs

uptightly

cross-modally

1. Neo-classical compounds astro-naut hydro-electric mechano-phobe

table 2: english compounds (bauer, 1983)

An alternative approach is to classify compounds in terms of the semantic relationship between the compound and its head. The head of a compound is the constituent modified by the compound's other constituents. In English, heads of compounds are typically the rightmost constituent (excluding any derivational and inflectional suffixes). For example, in *traffic-cop* the head is *cop*, which is modified by *traffic*; in *line-backer* the

head is *backer*, which is modified by *line*. Linguists distinguish at least three different semantic relations between the head and modifier(s) of compounds.

First, the compound represents a subtype of whatever the head represents. For instance, a *traffic-cop* is a kind of cop; a *teapot* is a kind of pot; a *fog-lamp* is a kind of lamp; a *blue-jay* is a kind of jay. That is, the head names the type, and the compound names the subtype. These are called **endocentric compounds**.

Second, the compound names a subtype, but the type is not represented by either the head or the modifier in the compound. For example, *Dead-head*, *redhead*, and *pickpocket* represent types of people by denoting some distinguishing characteristic. There is typically another word, not included in the compound, that represents the type of which the compound represents the subtype. In the case of *Deadhead*, *redhead*, and *pickpocket* this other word is *person*, so a *Deadhead* is a person who is an enthusiastic fan of the band *The Grateful Dead*. These are called **exocentric compounds**.

Third, there are compounds in which both elements are heads; each contributes equally to the meaning of the whole and neither is subordinate to the other, for instance, *bitter-sweet*. Compounds like these can be paraphrased as both X and Y, e.g., “bitter and sweet.” Other examples include *teacher-researcher* and *producer-director*. These can be called **coordinative compounds**.

As a third (and final) possible mode of analyzing compounds we briefly consider that used in the series of modern traditional grammars prepared by Quirk, Greenbaum, Leech and Svartvik (1972, 1985). In this method, the compounds are analyzed and classified according to the relationships among their constituents when the meaning of the compound is expressed as a phrase or clause. For example:

Phrases

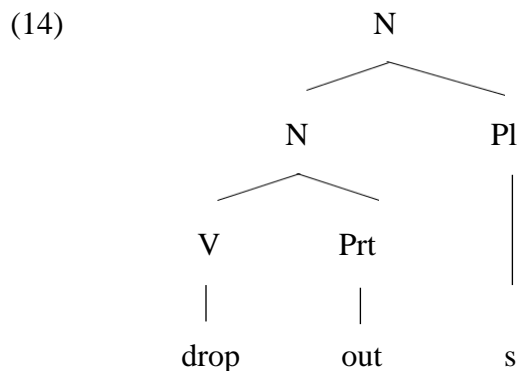
bee-sting a sting by a bee

blood-test	a test of blood
swimming pool	a pool for swimming
adding machine	a machine for adding
girlfriend	a friend who is a girl
killer shark	a shark which is a killer
windmill	a mill powered by wind
motorcycle	a cycle powered by a motor
self-control	someone able to control self

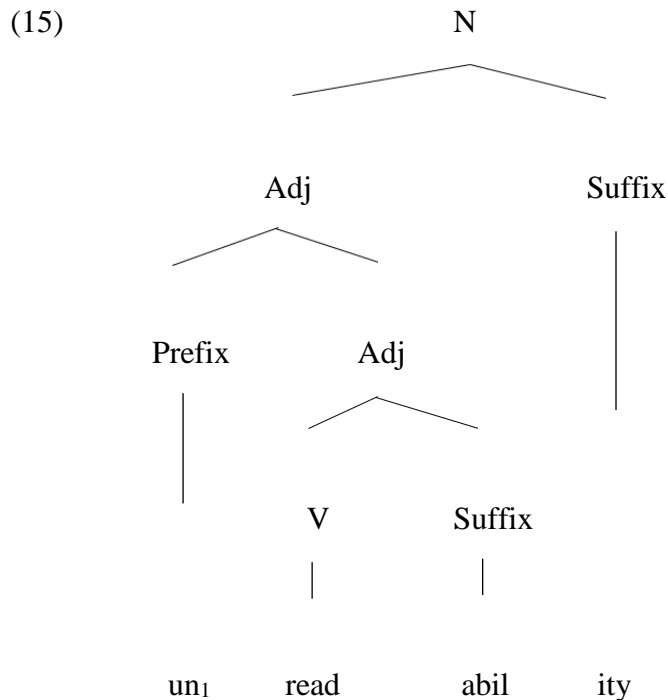
Clauses

sunrise	when the sun rises
---------	--------------------

table 3: underlying syntactic/semantic analysis of English *Morphology and Word Formation*



Consider another example: *unreadability*. We analyze this word as $[N[Adj]un_1[Adj[vread]abil]]ity]$, represented by the following tree:



Let's consider this analysis more closely. The suffix {-able} attaches to verbs to create adjectives. Besides *readable* we have the adjectives *doable*, *manage-able*, and *attachable*, which are derived from the verbs *read*, *do*, *manage*, and *attach*, respectively. We can represent this part of the word as: [Adj[vread] able].

The prefix {un₁-} attaches to adjectives, meaning “not” or “the converse of.” Compare *unwise*, *unfair*, *ungrateful*, *uncomfortable*, *unmanageable* with *unreadable*. All can be glossed as not having the quality denoted by the adjective to which they are attached: “not comfortable,” “not fair,” etc. This morpheme must be distinguished from the prefix {un₂-} meaning “to reverse the action,” which can be attached only to verbs (e.g., *untie*).

{Un₁-} cannot attach to the verb *read*; although there is the word *unread*, pronounced [ʌnrEd], not [ʌnrɪd], an adjective meaning “not read” and derived from the past participle of *read*. Consequently, in *unreadable*, {-able} must be attached to

{read} to create the adjective *readable*. {Un₁- } may then be attached to *readable* to create *unreadable*. We will represent this part of the word as: [Adj₁un₁[Adj₁[vread]able]].

The suffix {- ity} attaches to adjectives to create abstract nouns. Consequently it must be attached to the adjective *unreadable*. The structure of the entire word therefore must be: [N[Adj₁un₁[Adj₁[vread]able]]ity], as specified above. In pronunciation the morpheme {-able} will be assigned its allo-morph /@bIl/ (spelled <abil>, the same allomorph that appears in *ability*).

Classifying words by their morphological Properties

Once the morphemes of a language have been identified, their allomorphs determined, and their distributions specified, we can use our analysis to assign the words of a language to parts of speech. For many words, inflections provide the main basis of this assignment. Refer to Table 1 for the list of English inflections.

Nouns can be identified as those words that can be inflected for plural. Verbs are words that can be inflected for 3rd person singular present tense, past tense, past participle, and progressive. These forms are often re-

ferred to as the principal parts of the verb.

Short adjectives and adverbs are words that can be inflected for comparative and superlative.

Derivational regularities can also be used to classify words. We can, for example, classify as adverbs words derived from adjectives by the addition of the suffix {- ly}, e.g., *quickly*.

Classifying words on the basis of their internal morphological structure works only up to a point. There are lots of words that are not internally complex and so cannot be classified without recourse to other types of criteria. For example, the preposition *to* has

no internal morphological structure and so cannot be assigned to a grammatical class on that basis. Likewise, adverbs such as *hard* or *fast* lack the characteristic {- ly} ending. It becomes necessary to use other criteria to classify these and many other words. We consider in detail the principles which have been proposed for assigning words to parts of speech in the chapters on Major and Minor Parts of Speech in this book

on

Prefixes

Class/category changing

a- blaze	Adj < V
be- calm	V < Adj
be- friend	V < N
en- tomb	V < N

Class maintaining

Nouns

arch- monetarist

mal- nutrition

micro-dot

mini- dress

step- father

Verbs

de- escalate

Adjectives

a- typical

cis- lunar

extra- sensory

Noun or Verb

fore- ground

back-ground

mis- fortune

mis- lead

re- arrangement

Noun or Adjective

ex- President

ex- orbital

in- definite

mid- morning

mid- Victorian

Verb or Adjective

circum- navigate

circum- polar

Noun, Verb, or Adjective

co- author

counter- culture

counter- demonstrate

counter- intuitive

dis- ambiguate

dis- bound

dis- information

inter- mix

sub- conscious

sub- let

Suffixes

Creating Nouns

from Nouns

- dom	king- dom
- er	Birch- er
- ess	lion- ess
- ette	kitchen- ette
- iana	Victor- iana
- hood	man- hood
- ism	absentee- ism
- let	stream- let
- ling	duck- ling
- scape	sea- scape
- ship	kin- ship

from Verbs

- al	arriv- al
- ary	dispens- ary
- ation (esp. with - ize)	categor- iz- ation
- ee	blackmail- ee
- er	kill- er
- ment	manage- ment
- ure	clos- ure

from Adjectives

- ce	dependen- ce
- cy	excellen- cy
- dom	free- dom

- er	six- er
- hood	false- hood
- ist	social- ist
- ity	divin- ity
- ness	good- ness
- th	warm- th

Derived Verbs

from Nouns

- fy	metr- ify
- ize	Cambodian- ize

from Adjectives

- en	short- en
------	-----------

Derived Adjectives

from Nouns

- al	education- al (allomorphs/allographs: - ial, - ual: presidential, habitual)
- ate	passion- ate
- en	wood- en
- ese	Peking- ese
- esque	pictur- esque
- ful	doubt- ful
- ic	algebra-ic
- less	clue- less
- ly	friend- ly
- ous	venom- ous

- y catt- y

from Verbs

- able unbeliev- able
- less count- less
- ant/- ent absorb- ent
- atory affirm- atory
- ful resent- ful
- ive generat- ive

from Adjectives

- ish green- ish
- ly good- ly

Derived Adverbs

- ly slow- ly
- ward(s) in- ward(s)
- wise length- wise

Miscellaneous

down- er
iff- y, upp- itty
in- ness, much- ness, such- ness,
there- ness, why- ness
thus-ly

Morphology is the study of the way words are built up from smaller meaning bearing units, morphemes.

‘antiintellectualism’ -anti -ism -al -intellect

Free and bound morphemes

• intellect (free)

- anti- -ism, -al (bound)
- Stems and affixes • Complex words contain a central morpheme, which contributes the basic meaning, and a collection of other morphemes serving to modify this meaning in different ways. Morphology.
- **'disagreements'** agree (stem) dis- -ment -s (affixes) dis- prefix -ment suffix -s suffix
- English doesn't stack more than 4-5 affixes,
- Turkish 10 affixes.
- Agglutinative language. • Two broad classes of ways to form words from morphemes: inflection and derivation.
- **Inflection:** is the combination of a word stem with a grammatical morpheme, resulting in a word of the same class • cat-s cats play-ed
- **Derivation:** combination of a word stem with a grammatical morpheme, resulting in a word of a different class • agree -ment Morphology S.Ananiadou
- **English Inflectional Morphology** • English nouns have two kinds of inflection: plural & possessive • cat cats / ibis ibises / finch finches / box boxes • llama's / children's / llamas' • English verbal inflection is more complicated • main verbs (eat/sleep) • modal verbs (can / will/ should) • primary verbs (be, have, do) (see Quirk et al: Grammar of English Language) • Regular verbs (walk / walks / walking / walked) • Irregular verbs (eat / eats / eating / ate / eaten) Morphology S.Ananiadou
- **Derivational Morphology** • Syntactic category changing e.g. nominalization computerize ® computerization Suffix Base Noun/Verb/adjective Derived Noun -ation computerize computerization -ee appoint appointee -er kill killer -ness fuzzy fuzziness -al computation computational -able like likeable -less clue clueless Morphology S.Ananiadou
- **Derivation is less productive** • Affixes attach to stems and to each other according to certain constraints • Level Ordering in Derivation • In English we distinguish 2 types of affixation • class I affixation (+) • class II affixation (#) •

Class I occurs before class II I -> ion, ity, ate, ive, ic ... II -> y, ly, like, ful, ness, less, hood ... danger-ous1-ness2 *fear-less2-ity1 *tender-ness2-ous1 Morphology S.Ananiadou

- **Members of the same family may appear in any order with** respect to each other fear-less-ness tender-ness-less • Ordering Hypothesis about occurrence of morphological processes occur or morphotactics Class I affixation Class II affixation Inflection Compounding Morphology S.Ananiadou
- **Finite State Morphological Parsing** • Take an input like ‘cats’ and produce output forms like ‘cat +N +PL’ (morphological features) • In order to build a morphological parser we need: • lexicon • morphotactics • orthographic rules (spelling rules) model the changes occurring when two morphemes combine e.g. city ® cities How to use FSA to model morphotactic information FST as a way of modeling morphological features in the lexicon How to use FSTs to model orthographic rules Morphology S.Ananiadou
- **Lexicon and Morphotactics** • A lexicon is a repository of words • Since we cannot list every word in the language, computational lexicons are structured as a list of stems and affixes with a representation of the morphotactics. • One way to model morphotactics is the finite-state automaton Reg-noun Plural q0 q1 q2 Irregular-pl-noun Irreg-sg-noun Morphology S.Ananiadou
- **reg-noun irreg-pl-noun irreg-sg-noun** plural fox geese goose -s cat sheep sheep dog mice mouse aardvark reg-verb irreg-verb- irreg-past past past-part pres-part 3sg stem stem verb walk cut caught -ed -ed -ing -s fry speak ate talk sing eaten impeach sang spoken • English derivational morphology is more complex than inflectional morphology, automata for modeling are complex Morphology

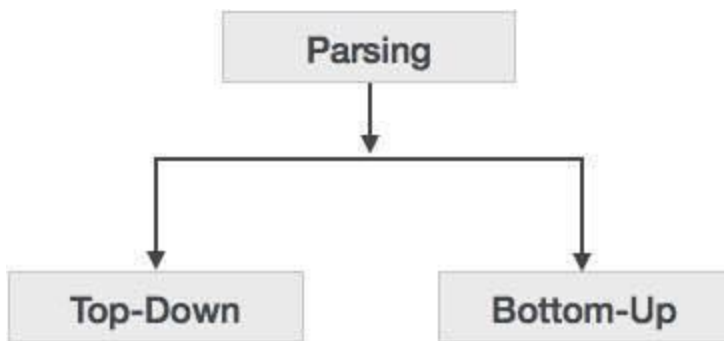
Morphological Parsing

There are 2 types of Parsing Technique present in parsing, first one is Top-down parsing and second one is Bottom-up parsing.

Top-down Parsing is a parsing technique that first looks at the highest level of the parse tree and works down the parse tree by using the rules of grammar while **Bottom-up Parsing** is a

parsing technique that first looks at the lowest level of the parse tree and works up the parse tree by using the rules of grammar.

Syntax analyzers follow production rules defined by means of context-free grammar. The way the production rules are implemented (derivation) divides parsing into two types : top-down parsing and bottom-up parsing.



Top-down Parsing

When the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input, it is called top-down parsing.

- **Recursive descent parsing** : It is a common form of top-down parsing. It is called recursive as it uses recursive procedures to process the input. Recursive descent parsing suffers from backtracking.
- **Backtracking** : It means, if one derivation of a production fails, the syntax analyzer restarts the process using different rules of same production. This technique may process the input string more than once to determine the right production.

Bottom-up Parsing

As the name suggests, bottom-up parsing starts with the input symbols and tries to construct the parse tree up to the start symbol.

Example:

Input string : $a + b * c$

Production rules:

```
S → E
E → E + T
E → E * T
E → T
T → id
```

Let us start bottom-up parsing

```
a + b * c
```

Read the input and check if any production matches with the input:

```
a + b * c
T + b * c
E + b * c
E + T * c
E * c
E * T
E
S
```

Bottom Up or Shift Reduce Parsers | Set 2

In this article, we are discussing the Bottom Up parser.

Bottom Up Parsers / Shift Reduce Parsers

Build the parse tree from leaves to root. Bottom-up parsing can be defined as an attempt to reduce the input string w to the start symbol of grammar by tracing out the rightmost derivations of w in reverse.

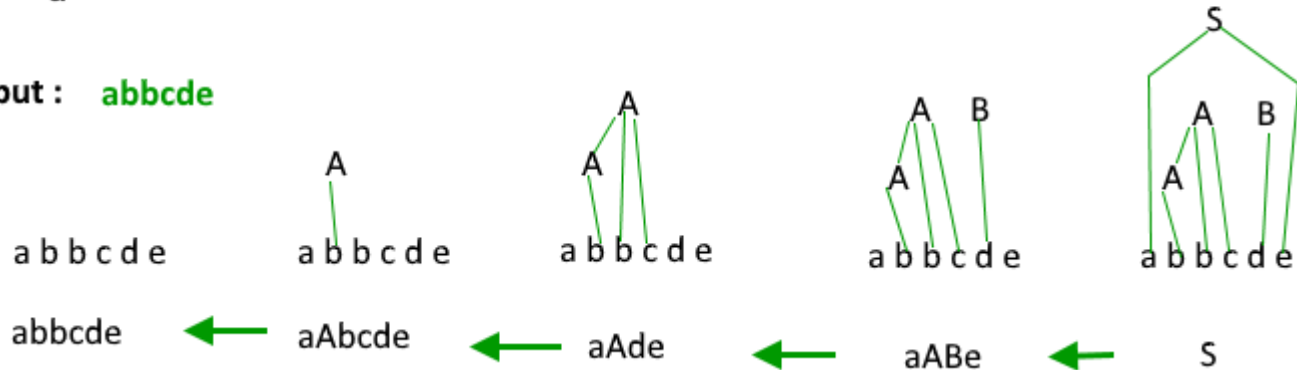
Eg.

$S \rightarrow aABe$

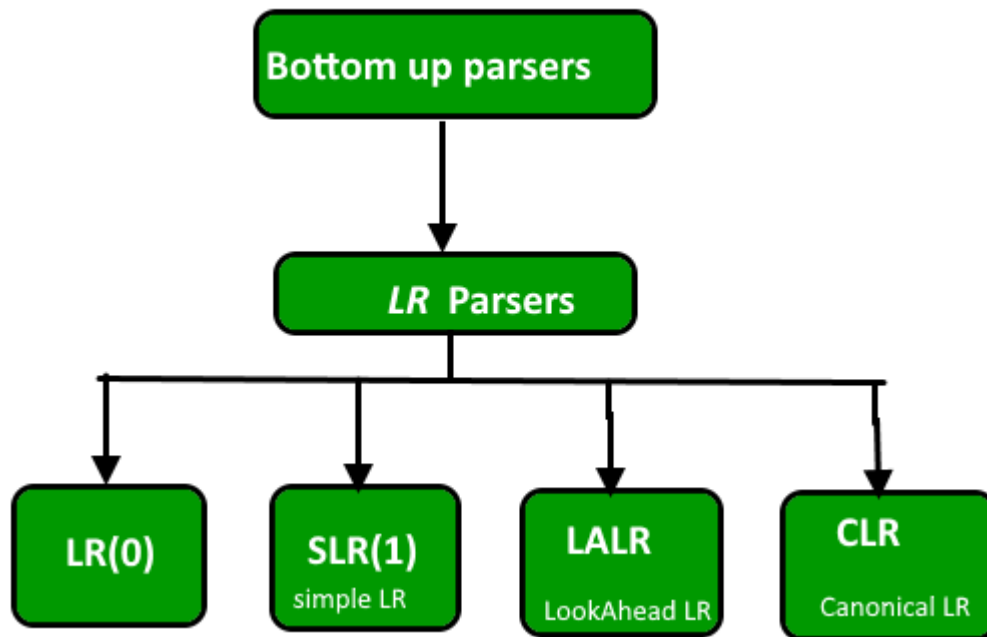
$A \rightarrow Abc/b$

$B \rightarrow d$

Input : **abbcd e**



Classification of bottom up parsers



A general shift reduce parsing is LR parsing. The L stands for scanning the input from left to right and R stands for constructing a rightmost derivation in reverse.

Benefits of LR parsing:

1. Many programming languages using some variations of an LR parser. It should be noted that C++ and Perl are exceptions to it.

2. LR Parser can be implemented very efficiently.

Bottom up Parsing The basic idea of bottom up parsing and recognition is: I to begin with the concrete data provided by the input string — that is, the words we have to parse/recognize — and try to build bigger and bigger pieces of structure using this information. I Eventually we hope to put all these pieces of structure together in a way that shows that we have found a sentence. Putting it another way, bottom up parsing is about moving from concrete low-level information to more abstract high-level information. This is reflected in a very obvious point about any bottom up algorithm: in bottom up parsing, we use our CFG rules right to left. Contents First Last Prev Next J 7.1. A bit more concretely Consider the CFG rule $C \rightarrow P1, P2, P3$. Working bottom up means that we will try to find a P1, a P2, and a P3 in the input that are right next to each other. If we find them, we will use this information to conclude that we have found a C. That is, in bottom up parsing, the flow of information is from the right hand side (P1, P2, P3) of the rules to the left hand side of the rules (C). Let's look at an example of bottom up parsing/ recognition start from a linguistics input. Contents First Last Prev Next J We have the rule $pn \rightarrow marcellus$, and this lets us build: $np\ tv\ pn$ 7. We also have the rule $np \rightarrow pn$ so using this right to left we build: $np\ tv\ np$ 8. Are there any more strings of length 1 we can rewrite using our context free rules right to left? No — we've done them all. 9. So now we start again at the beginning looking for strings of length 2 that we can rewrite using our CFG rules right to left. And there is one: we have the rule $vp \rightarrow tv\ np$, and this lets us build: $np\ vp$ 10. Are there any other strings of length 2 we can rewrite using our CFG rules right to left? Yes — we can now use: $s \rightarrow np\ vp$, we have built: s 11. And this means we are finished. Working bottom up we have succeeded in rewriting our original string of symbols into the symbol s — so we have successfully recognized “Vincent shot Marcellus” as a sentence. Contents First Last Prev Next J 7.3. Example Sara wears the new dress $pn \rightarrow sara\ pn$ wears the new dress $np \rightarrow pn\ np$ wears the new dress $tv \rightarrow wears\ np\ tv$ the new dress $det \rightarrow the\ np\ tv\ det$ new dress $adj \rightarrow new\ np\ tv\ det\ adj$ dress $n \rightarrow dress\ np\ tv\ det\ adj\ n$ $n \rightarrow adj\ n\ np\ tv\ det\ n$ $np \rightarrow det\ n\ np\ tv\ np\ vp \rightarrow tv\ np\ np\ vp\ s \rightarrow np\ vp\ s$ Contents First Last Prev Next J 7.4. Exercise Given the lexicon below, build the

CFG rules and use the same strategy described above to parse the input strings below. 1. “John saw the man with the telescope.” pn ---> john n ---> telescope tv ---> saw det ---> the n ---> park p ---> with n ---> man p ---> in How many parse trees do you obtain? s ---> np vp np ---> det n np ---> det n pp np --> pn vp ---> tv np Contents First Last Prev Next J vp ---> tv np pp pp ---> p np

There are some differences present to differentiate these two parsing techniques, which are given below:

S.NO	TOP DOWN PARSING	BOTTOM UP PARSING
1.	It is a parsing strategy that first looks at the highest level of the parse tree and works down the parse tree by using the rules of grammar.	It is a parsing strategy that first looks at the lowest level of the parse tree and works up the parse tree by using the rules of grammar.
2.	Top-down parsing attempts to find the left most derivations for an input string.	Bottom-up parsing can be defined as an attempts to reduce the input string to start symbol of a grammar.
3.	In this parsing technique we start parsing from top (start symbol of parse tree) to down (the leaf node of parse tree) in top-down manner.	In this parsing technique we start parsing from bottom (leaf node of parse tree) to up (the start symbol of parse tree) in bottom-up manner.
4.	This parsing technique uses Left Most Derivation.	This parsing technique uses Right Most Derivation.
5.	It’s main decision is to select what	It’s main decision is to select when

production rule to use in order to
construct the string.

to use a production rule to reduce
the string to get the starting
symbol.

The goal of *morphological parsing* is to find out what morphemes a given word is built from. For example, a morphological parser should be able to tell us that the word *cats* is the plural form of the noun stem *cat*, and that the word *mice* is the plural form of the noun stem *mouse*. So, given the string *cats* as input, a morphological parser should produce an output that looks similar to *cat N PL*. Here are some more examples:

mouse mouse N SG
mice mouse N PL
foxes fox N PL

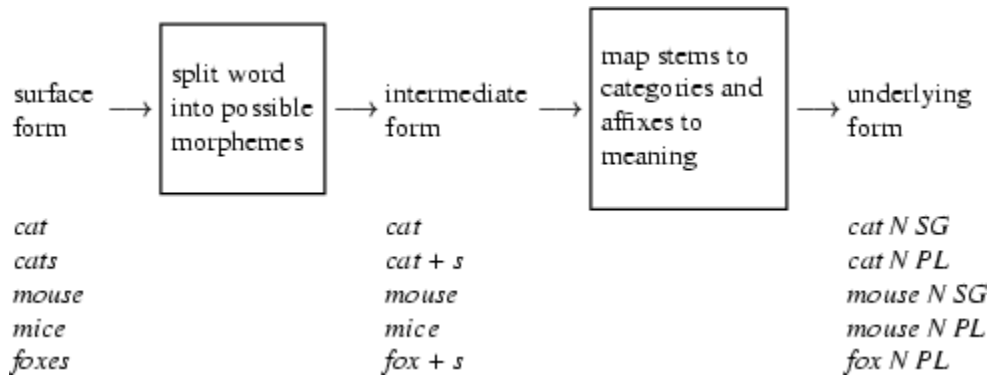
Morphological parsing yields information that is useful in many NLP applications. In parsing, e.g., it helps to know the agreement features of words. Similarly, grammar checkers need to know agreement information to detect such mistakes. But morphological information also helps spell checkers to decide whether something is a possible word or not, and in information retrieval it is used to search not only *cats*, if that's the user's input, but also for *cat*.

To get from the surface form of a word to its morphological analysis, we are going to proceed in two steps. First, we are going to split the words up into its possible components. So, we will make *cat + s* out of *cats*, using + to indicate morpheme boundaries. In this step, we will also take spelling rules into account, so that there are two possible ways of splitting up *foxes*, namely *foxe + s* and *fox + s*. The first one assumes that *foxe* is a stem and *s* the suffix, while the second one assumes that the stem is *fox* and that the *e* has been introduced due to the spelling rule that we saw above.

In the second step, we will use a lexicon of stems and affixes to look up the categories of the stems and the meaning of the affixes. So, *cat + s* will get mapped to *cat NP PL*, and *fox + s* to *fox N PL*. We will also find out now that *foxe* is not a legal stem. This tells us that splitting *foxes* into *foxe + s* was actually an incorrect way of

splitting *foxes*, which should be discarded. But note that for the word *houses* splitting it into *house + s* is correct.

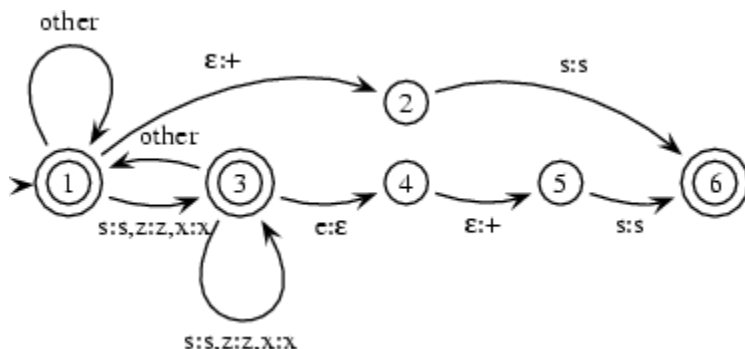
Here is a picture illustrating the two steps of our morphological parser with some examples.



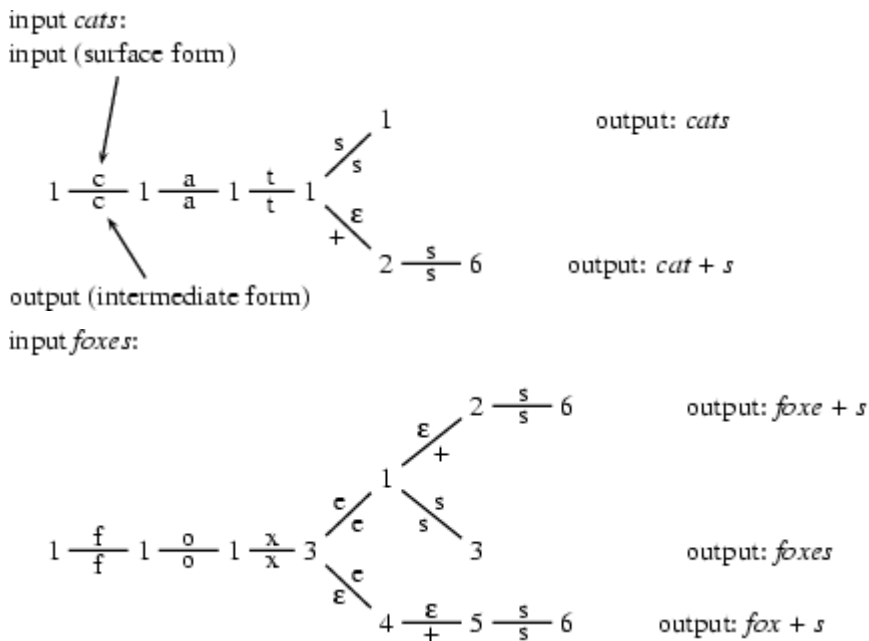
We will now build two transducers: one to do the mapping from the surface form to the intermediate form and the other one to do the mapping from the intermediate form to the underlying form.

From the Surface to the Intermediate Form

To do morphological parsing this transducer has to map from the surface form to the intermediate form. For now, we just want to cover the cases of English singular and plural nouns that we have seen above. This means that the transducer may or may not insert a morpheme boundary if the word ends in *s*. There may be singular words that end in *s* (e.g. *kiss*). That's why we don't want to make the insertion of a morpheme boundary obligatory. If the word ends in *ses*, *xes* or *zes*, it may furthermore delete the *e* when introducing a morpheme boundary. Here is a transducer that does this. The "other" arc in this transducer stands for a transition that maps all symbols except for *s*, *z*, *x* to themselves.



Let's see how this transducer deals with some of our examples. The following graphs show the possible sequences of states that the transducer can go through given the surface forms *cats* and *foxes* as input.



From the Intermediate Form to the Morphological Structure

Now, we want to take the intermediate form that we produced in the previous section and map it to the underlying form. The input that this transducer has to accept is of one of the following forms:

1. regular noun stem, e.g. *cat*
2. regular noun stem + *s*, e.g. *cat + s*
3. singular irregular noun stem, e.g. *mouse*
4. plural irregular noun stem, e.g. *mice*

In the first case, the transducer has to map all symbols of the stem to themselves and then output *N* and *SG*. In the second case, it maps all symbols of the stem to themselves, but then outputs *N* and replaces *PL* with *s*. In the third case, it does the same as in the first case. Finally, in the fourth case, the transducer should map the irregular plural noun stem to the corresponding singular stem (e.g. *mice* to *mouse*) and then it should add *N* and *PL*. So, the general structure of this transducer looks like this:

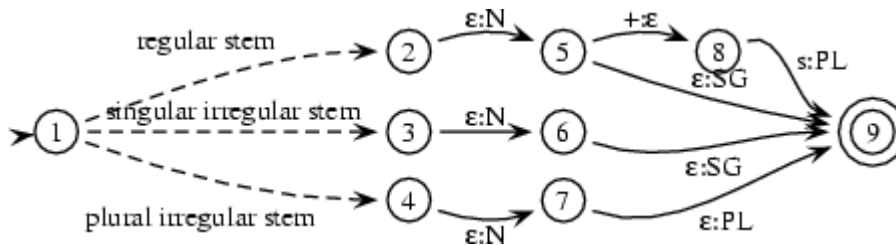
Top-down parsing constructs parse tree for the input string, starting from root node and creating the nodes of parse tree in pre-order.

It is done by leftmost derivation for an input string.

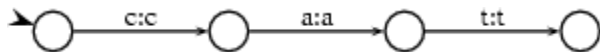
$$E \rightarrow E + E \mid E * E \mid id$$

Solution:

$$\begin{aligned}
 w &= id + id * id \\
 E &\xrightarrow{lm} E + E \\
 E &\xrightarrow{lm} id + E \quad [E \rightarrow id] \\
 E &\xrightarrow{lm} id + E * E \quad [E \rightarrow E * E] \\
 E &\xrightarrow{lm} id + id * E \quad [E \rightarrow id] \\
 E &\xrightarrow{lm} id + id * id \quad [E \rightarrow id]
 \end{aligned}$$



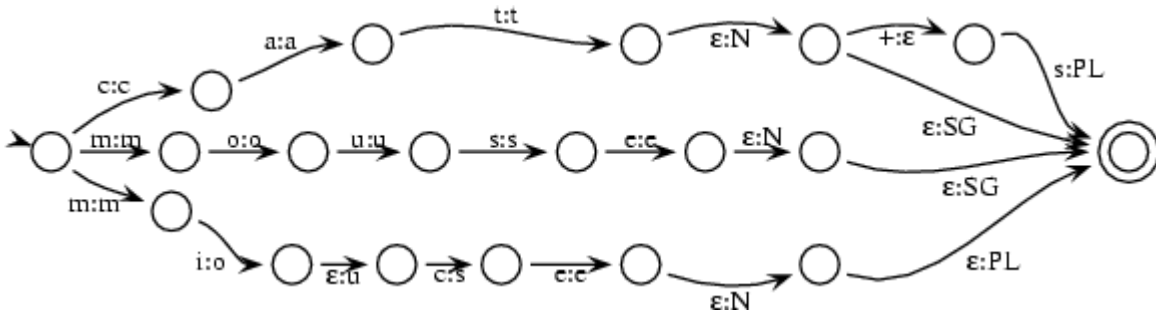
What still needs to be specified is how exactly the parts between state 1 and states 2,3, and 4 respectively look like. Here, we need to recognize noun stems and decide whether they are regular or not. We do this by encoding a lexicon in the following way. The transducer part that recognizes *cat*, for instance, looks like this:



And the transducer part mapping *mice* to *mouse* can be specified as follows:



Plugging these (partial) transducers into the transducer given above we get a transducer that checks that input has the right form and adds category and number information.



Combining the two Transducers

If we now let the two transducers for mapping from the surface to the intermediate form and for mapping from the intermediate to the underlying form run in a cascade (i.e. we let the second transducer run on the output of the first one), we can do a morphological parse of (some) English noun phrases. However, we can also use this transducer for generating a surface form from an underlying form. Remember that we can change the direction of translation when using a transducer in translation mode.

Now, consider the input *berries*. What will our cascaded transducers make out of it? The first one will return two possible splittings, *berries* and *berrie + s*, but the one that we would want, *berry + s*, is not one of them. The reason for this is that there is another spelling rule at work, here, which we haven't taken into account at all. This rule is saying that ``y changes to *ie* before *s*". So, in the first step there may be more than one spelling rules that all have to be applied.

There are basically two ways of dealing with this. First, we can formulate the transducers for each of the rules in such a way that they can be run in a cascade. Another possibility is to specify the transducers in such a way that they can be applied in parallel. There are algorithms for combining several cascaded transducers or several transducers that are supposed to be applied in parallel into a single transducer. However, these algorithms only work, if the individual transducers obey some restrictions so that we have to take some care when specifying them.

Putting it in Prolog

If you want to implement the small morphological parser that we have seen in the previous section, all you really have to do is to translate the transducer specifications into

the Prolog format that we used in the last lecture. Then, you can use last lecture's transducer program to let them run.

We won't show in detail what the transducers look like in Prolog, but we want to have a quick look at the *e* insertion transducer, because it has one interesting feature; namely, the *other* transition. How can we represent this in Prolog?

Here are all arcs going out of state 6 in Prolog notation, except for the *other* arc.

```
arc(6,2,z:z).
```

```
arc(6,2,s:s).
```

```
arc(6,2,x:x).
```

```
arc(6,3,'^':'^').
```

Now, the *other* arc should translate *any* symbol except for *z*, *s*, *x*, *^* to itself. In Prolog, we can express this using cuts and exploiting the fact that Prolog searches the database top down:

```
arc(6,2,z:z) :- !.
```

```
arc(6,2,s:s) :- !.
```

```
arc(6,2,x:x) :- !.
```

```
arc(6,3,'^':'^') :- !.
```

```
arc(6,1,X:X) :- !.
```

Top-down parsing

Top-down parsing in computer science is a parsing strategy where one first looks at the highest level of the parse tree and works down the parse tree by using the rewriting rules of a formal grammar. LL parsers are a type of parser that uses a top-down parsing strategy.

Top-down parsing is a strategy of analyzing unknown data relationships by hypothesizing general parse tree structures and then considering whether the known fundamental structures are compatible with the hypothesis. It occurs in the analysis of both natural languages and computer languages.

Top-down parsing can be viewed as an attempt to find left-most derivations of an input-stream by searching for parse-trees using a top-down expansion of the given formal grammar rules. Inclusive choice is used to accommodate ambiguity by expanding all alternative right-hand-sides of grammar rules.

Simple implementations of top-down parsing do not terminate for left-recursive grammars, and top-down parsing with backtracking may have exponential time complexity with respect to the length of the input for ambiguous CFGs.^[3] However, more sophisticated top-down parsers have been created by Frost, Hafiz, and Callaghan^{[4][5]} which do accommodate ambiguity and left recursion in polynomial time and which generate polynomial-sized representations of the potentially exponential number of parse trees.

Programming language application

A compiler parses input from a programming language to an internal representation by matching the incoming symbols to production rules. Production rules are commonly defined using Backus-Naur form. An LL parser is a type of parser that does top-down parsing by applying each production rule to the incoming symbols, working from the left-most symbol yielded on a production rule and then proceeding to the next production rule for each non-terminal symbol encountered. In this way the parsing starts on the Left of the result side (right side) of the production rule and evaluates non-terminals from the Left first and, thus, proceeds down the parse tree for each new non-terminal before continuing to the next symbol for a production rule.

For example:

- its string will be A=acdf

would match and attempt to match next. Then would be tried. As one may expect, some languages are more ambiguous than others. For a non-ambiguous language in which all productions for a non-terminal produce distinct strings: the string produced by one production will not start with the same symbol as the string produced by another production. A non-ambiguous language may be parsed by an LL(1) grammar

where the (1) signifies the parser reads ahead one token at a time. For an ambiguous language to be parsed by an LL parser.

The common solution to this problem is to use an LR parser, which is a type of shift-reduce parser, and does bottom-up parsing.

Accommodating left recursion in top-down parsing

A formal grammar that contains left recursion cannot be parsed by a naive recursive descent parser unless they are converted to a weakly equivalent right-recursive form. However, recent research demonstrates that it is possible to accommodate left-recursive grammars (along with all other forms of general CFGs) in a more sophisticated top-down parser by use of curtailment. A recognition algorithm which accommodates ambiguous grammars and curtails an ever-growing direct left-recursive parse by imposing depth restrictions with respect to input length and current input position, is described by Frost and Hafiz in 2006.^[6] That algorithm was extended to a complete parsing algorithm to accommodate indirect (by comparing previously computed context with current context) as well as direct left-recursion in polynomial time, and to generate compact polynomial-size representations of the potentially exponential number of parse trees for highly ambiguous grammars by Frost, Hafiz and Callaghan in 2007.^[4] The algorithm has since been implemented as a set of parser combinators written in the Haskell programming language. The implementation details of these new set of combinators can be found in a paper^[5] by the authors, which was presented in PADL'08. The X-SAIGA site has more about the algorithms and implementation details.

Time and space complexity of top-down parsing

When top-down parser tries to parse an ambiguous input with respect to an ambiguous CFG, it may need exponential number of steps (with respect to the length of the input) to try all alternatives of the CFG in order to produce all possible parse trees, which eventually would require exponential memory space. The problem of exponential time complexity in top-down parsers constructed as sets of mutually recursive functions has been solved by Norvig in 1991.^[7] His technique is similar to the use of dynamic

programming and state-sets in Earley's algorithm (1970), and tables in the CYK algorithm of Cocke, Younger and Kasami.

The key idea is to store results of applying a parser p at position j in a memorabile and to reuse results whenever the same situation arises. Frost, Hafiz and Callaghan^{[4][5]} also use memoization for refraining redundant computations to accommodate any form of CFG in polynomial time ($\Theta(n^4)$ for left-recursive grammars and $\Theta(n^3)$ for non left-recursive grammars). Their top-down parsing algorithm also requires polynomial space for potentially exponential ambiguous parse trees by 'compact representation' and 'local ambiguities grouping'. Their compact representation is comparable with Tomita's compact representation of bottom-up parsing.^[8]

Using PEG's, another representation of grammars, packrat parsers provide an elegant and powerful parsing algorithm. See Parsing expression gr

1. Bottom up parsing
2. Bottom-up parsing is more general than top- down. ... Bottom-Up Parsing
3. Bottom-up Parsing • Two types of bottom-up parsing: • Operator-Precedence parsing • LR parsing • covers wide range of grammars. ...
4. Identify β in str such that $A \rightarrow \beta$ is ... repeat Str input string of terminals LR parsing reduces a string to the start symbol by inverting productions: Bottom-Up Parsing

Syntax

Introduction to Generalized Phrase Structure Grammar (GPSG)

Generalized phrase structure grammar (GPSG) is a framework for describing the syntax and semantics of natural languages. ... Constraint based grammars are based around defining certain syntactic processes as ungrammatical for a given language and assuming everything not thus dismissed is grammatical within that language.

Phrase structure grammar is a type of generative grammar in which constituent structures are represented by *phrase structure rules* or *rewrite rules*. Some of the different versions of phrase structure grammar (including *head-driven phrase structure grammar*) are considered in examples and observations below.

A phrase structure (or constituent) functions as the base component in the classic form of transformational grammar introduced by Noam Chomsky in the late 1950s. Since the mid-1980s, however, *lexical-function grammar* (LFG), *categorial grammar* (CG), and *head-driven phrase structure grammar* (HPSG) "have developed into well-worked-out alternatives to transformational grammar"

Examples and Observations

- "The underlying structure of a sentence or a phrase is sometimes called its *phrase structure* or *phrase marker*. . . . Phrase-structure rules provide us with the underlying syntactic structure of sentences we both produce and comprehend. . . .
- "There are different types of phrase-structure grammar. Context-free grammars contain only rules that are not specified for particular contexts, whereas context-sensitive grammars can have rules that can only be applied in certain circumstances. In a context-free rule, the left-hand symbol can always be rewritten by the right-hand one regardless of the context in which it occurs. For example, the writing of a verb in its singular or plural form depends on the context of the preceding noun phrase."

Rewrite Rules

"The idea of a PSG [phrase structure grammar] is simple. We first note what syntactic categories appear to exist in a given language, and what different internal structures each of these can have. Then, for each such structure, we write a rule that displays that structure. So, for example, an English sentence typically consists of a noun phrase followed by a verb phrase (as in *My sister bought a car*), and we, therefore, write a *phrase-structure rule* as follows:

$$S \rightarrow NP VP$$

This says that a sentence may consist of a noun phrase followed by a verb phrase. . . . We continue in this way until we have a rule for every structure in the language.

"Now the set of rules can be used to *generate* sentences. Starting with *S* (for 'sentence'), we apply some suitable rule to tell us what units the sentence consists of, and then to each of those units we apply a further rule to tell us what units *it* consists of, and so on."

"A phrase structure grammar consists of a set of ordered rules known as *rewrite rules*, which are applied stepwise. A rewrite rule has a single symbol on the left and one or more symbols on the right:

$$A \rightarrow B+C$$
$$C \rightarrow D$$

More than one symbol on the right constitutes a *string*. The arrow is read as 'is rewritten as,' 'has as its constituents,' 'consists of,' or 'is expanded as.' The plus sign is read as 'followed by,' but it is often omitted. The rule may also be depicted in the form of a tree diagram...

"The phrase structure rules also allow for choices. The optional choices are indicated with parentheses:

$$A \rightarrow (B)C$$

This rule reads that *A* is expanded as optionally *B* and obligatorily *C*. In every rewrite rule, at least one element must be obligatory. There may also be mutually exclusive choices of elements in a string; these are indicated with curly braces:

$$A \rightarrow \{B,C\}$$

This rule states that if you choose *B*, you can't choose *C*, but you must choose one—either *B* or *C*, but not both. Whether the mutually exclusive items are written on one line separated by commas or on separate lines does not matter, as long as they occur within braces."

Definite clause grammar (DCG)

Definite clause grammar (DCG) is a way of expressing grammar, either for natural or formal languages, in a logic programming language such as Prolog. It is closely related to the concept of attribute grammars / affix grammars from which Prolog was originally developed.

Definite Clause Grammars (DCGs) are convenient ways to represent grammatical relationships for various parsing applications. They can be used for natural language work, for creating formal command and programming languages.

For example, DCG is an excellent tool for parsing and creating tagged input and output streams, such as HTML or XML. The index and table of contents in this documentation are generated by a Prolog program that uses DCG to parse the HTML, looking for headings and index entries.

Definite Clause Grammars Examples

A simple example of parsing using DCG

```
sentence --> noun_phrase, verb_phrase.
```

```
noun_phrase --> determiner, noun.
```

```
verb_phrase --> verb, noun_phrase.
```

```
verb_phrase --> verb, sentence.
```

```
determiner --> [the].
```

```
determiner --> [a].
```

```
noun --> [cat].
```

```
noun --> [mouse].
```

```
verb --> [scares].
```

```
verb --> [hates].
```

```
?-sentence([the,cat,scares,the,mouse],[]).
```

Notice that the nonterminal symbol 'sentence' has no arguments when used in a DCG rule, but it is called with two arguments when used for parsing. The two arguments are: the sentence to be parsed, and an empty list. When DCG rules are translated from DCG to

regular Prolog two arguments are added to each of the nonterminal symbols. This explains why they need to be called with two extra arguments.

Parsing with grammatical agreement (singular/plural)

```
% The argument Number is the number of the subject and main verb.
```

```
% It is instantiated to 'singular' or 'plural'.
```

```
sentence(Number) --> noun_phrase(Number), verb_phrase(Number).
```

```
noun_phrase(Number) --> determiner(Number), noun(Number).
```

```
verb_phrase(Number) --> verb(Number), noun_phrase(_).
```

```
determiner(singular) --> [a].
```

```
determiner(_) --> [the].
```

```
determiner(plural) --> [].
```

```
noun(singular) --> [cat];[man];[mouse].
```

```
noun(plural) --> [cats];[men];[mice].
```

```
verb(singular) --> [scares];[hates].
```

```
verb(plural) --> [scare];[hate].
```

```
?-sentence(plural,[men,hate,mice],[]).
```

```
?-sentence(_,[the,men,hate,mice],[]).
```

```
?-sentence(_,[the,mice,scare,the,man],[]).
```

Including morphology in parsing

```
% From the book PROLOG PROGRAMMING IN DEPTH
```

```
% by Michael A. Covington, Donald Nute, and Andre Vellino
```

```
% (Prentice Hall, 1997). Copyright 1997 Prentice-Hall, Inc.
```

```
% For educational use only
```

```
sentence --> noun_phrase(N), verb_phrase(N).
```

```
noun_phrase(N) --> determiner(N), noun(N).
```

```
verb_phrase(N) --> verb(N), noun_phrase(_).
```

verb_phrase(N) --> verb(N), sentence.

determiner(singular) --> [a].

determiner(_) --> [the].

determiner(plural) --> [].

noun(N) --> [X], { morph(noun(N),X) }.

verb(N) --> [X], { morph(verb(N),X) }.

% morph(-Type,+Word)

% succeeds if Word is a word-form of the specified type.

morph(noun(singular),dog). % Singular nouns

morph(noun(singular),cat).

morph(noun(singular),boy).

morph(noun(singular),girl).

morph(noun(singular),child).

morph(noun(plural),children). % Irregular plural nouns

morph(noun(plural),X) :- % Rule for regular plural nouns

remove_s(X,Y),

morph(noun(singular),Y).

morph(verb(plural),chase). % Plural verbs

morph(verb(plural),see).

morph(verb(plural),say).

morph(verb(plural),believe).

morph(verb(singular),X) :- % Rule for singular verbs

remove_s(X,Y),

morph(verb(plural),Y).

```

% remove_s(+X,-X1) [lifted from TEMPLATE.PL]
% removes final S from X giving X1,
% or fails if X does not end in S.

remove_s(X,X1):-
    name(X,XList),
    remove_s_list(XList,X1List),
    name(X1,X1List).

remove_s_list("s",[]).

remove_s_list([Head|Tail],[Head|NewTail]) :-
    remove_s_list(Tail,NewTail).
Parsing and constructing the parse tree
/* this generates a parse tree for a simple English grammar */

sentence(sentence(X,Y)) -->
    noun_phrase(X), verb_phrase(Y).

noun_phrase(noun_phrase(X,Y)) -->
    determiner(X), noun(Y).

verb_phrase(verb_phrase(X,Y)) -->
    verb(X), noun_phrase(Y).

determiner(determiner(the)) --> [the].
determiner(determiner(a)) --> [a].
noun(noun(mouse)) --> [mouse].
noun(noun(cat)) --> [cat].
verb(verb(hate)) --> [hated].

```

verb(verb(scare)) --> [scared].

?-sentence(Parsetree,[the,mouse,hated,the,cat],[[]]).

/* this is a more complex grammar, which also checks for singular/plural
and returns the parse tree.

From Pereira and Warren paper, AI journal, 1980 */

sentence(s(NP,VP)) -->

noun_phrase(N, NP), verb_phrase(N,VP).

noun_phrase(N,np(Det,Noun,Rel)) -->

determiner(N, Det), noun(N,Noun), rel_clause(N,Rel).

noun_phrase(singular,np(Name)) -->

name(Name).

verb_phrase(N,vp(TV,NP)) -->

trans_verb(N,TV), noun_phrase(_,NP).

verb_phrase(N,vp(IV)) -->

intrans_verb(N,IV).

rel_clause(N,rel(that,VP)) -->

[that],verb_phrase(N,VP).

rel_clause(_,rel(nil)) --> [].

determiner(N,det(W)) --> [W],{is_determiner(W,N)}.

determiner(plural,det(nil)) --> [].

noun(N,n(Root)) --> [W],{is_noun(W,N,Root)}.

name(name(W)) --> [W],{is_name(W)}.

trans_verb(N,tv(Root)) --> [W],{is_trans(W,N,Root)}.

intrans_verb(N,iv(Root)) --> [W],{is_intrans(W,N,Root)}.

is_determiner(every,singular).

is_determiner(all,plural).

is_noun(man,singular,man).

is_noun(men,plural,men).

is_name(mary).

is_trans(likes,singular,like).

is_trans(like,plural,like).

Lexical functional grammar (LFG)

Lexical functional grammar (LFG) is a constraint-based grammar framework in theoretical linguistics. It posits two separate levels of syntactic structure, a phrase structure grammar representation of word order and constituency, and a representation of grammatical functions such as subject and object, similar to dependency grammar. The development of the theory was initiated by Joan Bresnan and Ronald Kaplan in the 1970s, in reaction to the theory of transformational grammar which was current in the late 1970s. It mainly focuses on syntax, including its relation with morphology and semantics. There has been little LFG work on phonology (although ideas from optimality theory have recently been popular in LFG research).

LFG views language as being made up of multiple dimensions of structure. Each of these dimensions is represented as a distinct structure with its own rules, concepts, and form. The primary structures that have figured in LFG research are:

- the representation of grammatical functions (f-structure). See feature structure.
- the structure of syntactic constituents (c-structure). See phrase structure rules, ID/LP grammar.

For example, in the sentence *The old woman eats the falafel*, the c-structure analysis is that this is a sentence which is made up of two pieces, a noun phrase (NP) and a verb phrase (VP). The VP is itself made up of two pieces, a verb (V) and another NP. The NPs are also analyzed into their parts. Finally, the bottom of the structure is composed of the words out of which the sentence is constructed. The f-structure analysis, on the other hand, treats the sentence as being composed of attributes, which include features such as number and tense or functional units such as subject, predicate, or object.

There are other structures which are hypothesized in LFG work:

- argument structure (a-structure), a level which represents the number of arguments for a predicate and some aspects of the lexical semantics of these arguments. See theta-role.
- semantic structure (s-structure), a level which represents the meaning of phrases and sentences. See Glue Semantics.
- information structure (i-structure)
- morphological structure (m-structure)
- phonological structure (p-structure)

The various structures can be said to be mutually constraining.

The LFG conception of linguistic structure differs from Chomskyan theories, which have always involved separate levels of constituent structure representation mapped onto each other sequentially, via transformations. The LFG approach has had particular success with nonconfigurational languages, languages in which the relation between structure and function is less direct than it is in languages like English; for this reason LFG's adherents consider it a more plausible universal model of language.

Another feature of LFG is that grammatical-function changing operations like passivization are relations between word forms rather than sentences. This means that the active-passive relation, for example, is a relation between two types of verb rather than two trees. Active and passive verbs involve alternative mapping of the participants to grammatical functions.

Through the positing of productive processes in the lexicon and the separation of structure and function, LFG is able to account for syntactic patterns without the use of transformations defined over syntactic structure. For example, in a sentence like *What did you see?*, where *what* is understood as the object of *see*, transformational grammar puts *what* after *see* (the usual position for objects) in "deep structure", and then moves it. LFG analyzes *what* as having two functions: question-focus and object. It occupies the position associated in English with the question-focus function, and the constraints of the language allow it to take on the object function as well.

A central goal in LFG research is to create a model of grammar with a depth which appeals to linguists while at the same time being efficiently parsable and having the rigidity of formalism which computational linguists require. Because of this,

computational parsers have been developed and LFG has also been used as the theoretical basis of various machine translation tools, such as AppTek's TranSphere, and the Julietta Research Group's Lekta.

Head-Driven Phrase Structure Grammar (HPSG)

- "Head-driven phrase structure grammar (HPSG) has evolved as a synthesis of ideas from a number of theoretical sources, including generalized phrase structure grammar (GPSG), categorial grammar, and formal theories of data structure representation . . . HPSG uses a fundamental theoretical strategy made familiar by GPSG: the enumeration of a class of objects, corresponding to expressions of some natural language, and a set of constraints whose interaction enforces the appropriate covariation of formal properties reflecting the dependencies that any grammar of that language must capture."
- "A head-driven phrase structure grammar of some language defines the set of signs (form/meaning/correspondences) which that language comprises. The formal entities that model signs in HPSG are complex objects called *feature structures*, whose form is limited by a set of constraints--some universal and some language parochial. The interaction of these constraints defines the grammatical structure of each such sign and the morphosyntactic dependencies which hold between its subcomponents. Given a specific set of such constraints, and a lexicon providing at least one feature structure description for each word in the language, an infinite number of signs is recursively characterized."

Lexicalized tree-adjoining grammars (LTAG)

Lexicalized tree-adjoining grammars (LTAG) are a variant of TAG in which each elementary tree (initial or auxiliary) is associated with a lexical item. A lexicalized grammar for English has been developed by the XTAG Research Group of the Institute for Research in Cognitive Science at the University of Pennsylvania.

Introduction to Tree Adjoining Grammars A Tree Adjoining Grammar (TAG) as defined traditionally is said to be specified by a finite set of elementary trees. Unlike the string rewriting formalisms that write recursion into the rules that generate the phrase structure, a TAG factors recursion and dependencies into a finite set of elementary trees. The elementary trees in a TAG correspond to minimal linguistic structures that localize the dependencies such as subcategorization, and filler-gap. There are two kinds of elementary trees: initial trees and auxiliary trees.

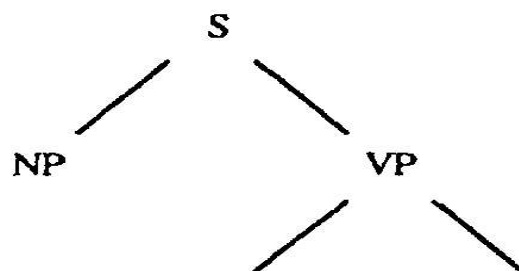
Originally, initial trees (e.g., c~1 and OL 2 in Figure 1) were defined to correspond to minimal sentential structures. Therefore, the root of an initial tree was required to be labeled by the symbol S. With the advent of lexicalized TAG and the use of the substitution operation, this assumption is no longer made (see c~3).

Auxiliary trees (ill, t2 in Figure 2) are usually defined to correspond to minimal recursive constructions. Thus, if the root of an auxiliary tree is labeled by a nonterminal 483 Computational Linguistics Volume 18, Number 4 f12 : S w A v VP* adv vo s* Figure 2 Auxiliary trees. symbol, A, then there is a distinguished node, called the foot node, in the frontier of this tree that is also labeled by A. The foot nodes of auxiliary trees, fll and /32, are indicated with an asterisk.

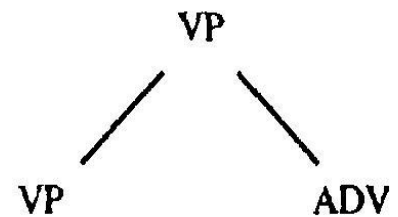
The adjoining operation is used to compose trees. An auxiliary tree, whose root and foot node are labeled A, can be adjoined at a node that is also labeled A. Adjoining may be described as follows: the subtree below the node of adjunction is excised; the auxiliary tree is inserted in its place; and the excised subtree is substituted at the foot node of the inserted auxiliary tree.

Tree-Adjoining Grammar

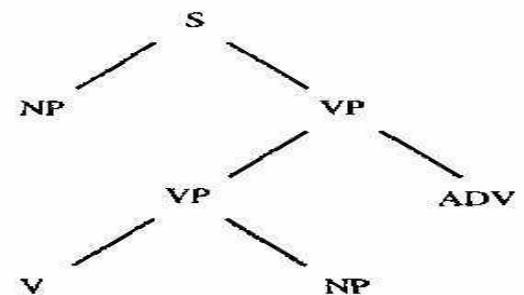
Another approach to handling unbounded dependencies is tree-adjoining grammars (TAGs) (Joshi, 1985). There are no grammar rules in this formalism. Rather, there is a set of initial tree structures that describe the simplest sentences of the language, and a tree operation, called adjoining, that inserts one tree into another to create a more complex structure. For example, a simple initial tree is



More complex sentences are derived using auxiliary trees, which capture the minimal forms of recursion in the language. For example, an auxiliary tree for allowing an adverbial in a verb phrase could be



The adjunction operation involves inserting an auxiliary tree-capturing recursion of some constituent C into another tree that contains a constituent C. Adjoining the auxiliary tree for VP above into the initial tree for S produces the new tree



By basing the formalism on trees, enough context is provided to capture long-distance dependencies. In this theory there is no movement. Instead the constituents start off being close to each other, and then additional structure is inserted between them by the adjoining operation. The result is a formalism of slightly greater power than CFGs but definitely weaker than context-sensitive grammars. The work on handling movement using gap threading and the definition of extraposition grammars can be found in Pereira (1981). Similar techniques can be found in many current parsers (such as Alshawi, 1992).

Augmented transition network

An augmented transition network or ATN is a type of graph theoretic structure used in the operational definition of formal languages, used especially in parsing relatively complex natural languages, and having wide application in artificial intelligence. An ATN can, theoretically, analyze the structure of any sentence, however complicated. ATN are modified transition networks and an extension of RTNs.

ATNs build on the idea of using finite state machines (Markov model) to parse sentences. W. A. Woods in "Transition Network Grammars for Natural Language Analysis" claims that by adding a recursive mechanism to a finite state model, parsing can be achieved much more efficiently. Instead of building an automaton for a particular sentence, a collection of transition graphs are built. A grammatically correct sentence is parsed by reaching a final state in any state graph. Transitions between these graphs are simply subroutine calls from one state to any initial state on any graph in the network. A sentence is determined to be grammatically correct if a final state is reached by the last word in the sentence.

This model meets many of the goals set forth by the nature of language in that it captures the regularities of the language. That is, if there is a process that operates in a number of environments, the grammar should encapsulate the process in a single structure. Such encapsulation not only simplifies the grammar, but has the added bonus of efficiency of operation. Another advantage of such a model is the ability to postpone decisions. Many grammars use guessing when an ambiguity comes up. This means that not enough is yet known about the sentence. By the use of recursion, ATNs solve this inefficiency by postponing decisions until more is known about a sentence.

UNIT-3

Semantics and Knowledge Representation

Computational Semantics

Semantics is the study of meaning • Meaning is the core of human communication. It is the msg that we want to convey (explicitly or implicitly) • Meaning representations are formal structures • Meaning representation languages are frameworks that specify the syntax and semantics of these representations.

Computational Semantics vs Pragmatics • Roughly, semantics is the meaning that can be deduced directly from an expression, with no extra-- linguistic information. –cf: "the sun is rising" vs "the bus" • Computational Semantics focuses not only on the abstract accounts of meanings, but also in a concrete formalizations that can support implementation.

Semantic Analysis... ... is the process that we use to –create representations of meaning –assign them to linguistic inputs.

The Representation of Meaning • Meaning of linguistic expressions can be captured in formal structures that we call meaning representations. • What we need are representation that bridge the gap from linguistic inputs to the non linguistic knowledge of the world • It requires access to the representations that link the linguistic elements involved in the task to the non-- linguistic 'knowledge of the world' needed to perform the task.

The aim of computational semantics is to find techniques for automatically constructing semantic representations for expressions of human language, representations that can be used to perform inference and verification of the given information. Computational semantics is potentially useful in such applications as information retrieval, information extraction, dialogue systems, question answering, interpreting controlled languages and so on.

Representation and Semantics

A representation is a symbolic structure in a system that can be understood to encode specified information—to carry meaning—because of mechanical processes that produce the representation as a function of appropriate states of the world and use the representation to guide the system’s behavior in appropriate ways. See Newell (1982), Fodor (1987), Gallistel & King (2009), Smith (to appear). Computer scientists’ interests in the meanings of representations largely parallel semanticists’ interests in the meanings of linguistic expressions. Meaning in language, of course, brings distinctive problems. An instructive way of foregrounding these problems is through the development of an explicit theory that characterizes the truth conditions of sentences in a fragment of natural language. See Davidson (1967), Larson & Segal (1995). We assume that we have an agreed metalanguage for which meaning is unproblematic (or outside the scope of our investigation).

The study of meaning I the conditions under which declarative sentences are true
I Truth conditions underlie the meanings of non-declarative sentences I questions — is
the corresponding declarative sentence true? What objects provide true and complete
answers? I imperatives — make the corresponding declarative sentence true I

The following tasks come under the purview of computational semantics:

Representation of knowledge of language:

Relations between words as specified in lexical knowledge base such as
WordNet, FrameNet, VerbNet and so on.

Syntactic structure, meaning representation and meaning
composition procedures Representation of knowledge of the world:

What are the objects that we refer to, how they are related, what are their
properties as in Semantic Ontologies such as ConceptNet, SUMO and so on.

Given a representation and a world (knowledge base of entities, their properties and relation between entities) what new conclusions – bits of meaning – can we infer?

This module presents the meaning representation and procedures for meaning composition. We will organize the module in the following sections. The next section describes the computational desiderata of semantic representation. Section 3 presents Neo-Davidsonian approach for semantic representation and its representation as *typed feature structure*. Section 4 presents “unification based approach” for compositionally computing meaning of larger expressions from the given lexicon and constructions licensed by the grammar of the language.

Computational Significance of Representation

As pointed out in Martin and Jurafsky (2009) and Copestake (2005), following are the basic requirements that a meaning representation must fulfill:

Unambiguous Representation: Natural languages are full of ambiguity. However, since one acts upon and make inferences on the basis of the meaning representation, such representations are required to be devoid of ambiguity. For example, a prepositional phrase (PP) can be a modifier of a noun or a verb. In (1), the PP modifies the verb because, the object noun and the noun in PP are not semantically compatible which is the case in (2).

- (1) Amit eats fish with salad.
- (2) Amit eats fish with a fork.

Now, the semantic representation of these two sentences are required to be such that the scope for ambiguity does not remain. We will formalize a representation in the next section where we will see how such ambiguity can be resolved in terms of semantic representation.

In cases where the meaning cannot be determined at the level of the sentence and the context beyond this sentence is necessary for deciding the meaning of the sentence, the representation strategies remain underspecified (Copestake 2005). The formalisms of Minimal Recursion Semantics (Copestake et al. 2001, 2005) very efficiently represent underspecification.

Expressiveness

Expressiveness: For the sake of brevity, there exists a tendency of dropping information (ellipsis) which are retrievable in natural language sentences. For example, in the following sentence we do not explicitly specify the experiencer of hunger because the information is retrievable:

(3) □□□ □□-□□□□□□□□□□□□ □□□ □□-□□□ □□□
 hunger feeling I food eat take-hab 1p pr
 “ I eat when I feel hungry”

An expressive semantic representation system will explicitly specify the experiencer information and will also convey that the experiencer and the doer of the verb 'eat' are the same individual. Otherwise, inferencing will become difficult.

Verifiability

Verifiability: The meaning representation will be such that it should be possible to determine the relationships between the meaning of a sentence and the world we know. Here the world can be seen as a knowledge base that consists of some facts about the real world. For example, we can represent the following facts in terms of relations:

- Is_A(Singh Junior College, College)
- Is_A(Bethun, College)
- Is_A(Sundari Devi, College)
- Teach(Bethun, Commerce)
- Teach(Singh Junior College, Science)

Teach(Bethun, Arts)

Teach(Bethun, Fine Arts)

These facts present names of some colleges and different programs that the colleges offer. Given the above world, the system can verify the state-of-affair described by a sentence representation with respect to the knowledge base. For example, the verification process will return an affirmative answer for the query: “Does Bethun College teach Arts?” Note that the query can be segmented in terms of the following two meaning representations:

(a) Is_A(Bethun, College)

(b) Teach(Bethun, Arts)

This kind of representation has the feature of Computational Tractability, one of the desirable characteristic features as maintained by Copestake (2005). She states that “It must be possible to process meanings and to check semantic equivalence and to express relationships between semantic representations straightforwardly.”

Canonicalness

Canonicalness: One can convey an information in many different ways in natural languages. For example, all the following sentences have identical meaning:

(4) The town is called Anantapur which has been developed in the last ten years.

(5) The town of Anantapur has been developed in the last ten years.

(6) The name of the town is Anantapur and it has been developed in the last ten years.

These sentences will have one canonical representation so that one can deduce that all these syntactically varied sentences have the same meaning. This becomes important for knowledge extraction and question answering tasks.

A Framework for Meaning Representation

There exist different formalisms for representing meaning especially in formal semantics (Dowty et al. 1981; Montague 1973). However, for computational purpose, it has been found that some variant of first order predicate logic is mostly preferred as a representational system because the process of verification and inferencing becomes efficient (Bos 2011). We will present here an event based flat semantic structure where *event* is given the status of an entity which can be quantified and related to other entities in terms of a defined set of relation. For example, the event semantic representation of the sentence in (4) will be something like given in (5) (ignoring tense for the time being):

(4) □□□□ □□□□□ □□ □□□ □□□ □□□□ □□

"Ram is eating curd with a spoon" Ram spoon with
curd eat 3pr cont

(6) $\exists e x y (\text{Is_A}(e, \text{Eating}), \text{agt}^1(e, \text{Ram}), \text{Is_A}(x, \text{Spoon}), \text{Is_A}(y, \text{Curd}), \text{thm}(y, \text{Curd}), \text{inst}(e, \text{Spoon}))$

We can read the above notation like in the following way. The expression $\text{Is_A}(x, \text{Spoon})$ entails there exists a predicate Is_A and x is a Spoon. The second argument of all predicates represent a concept. The motivation for taking event as an entity and quantifying over it was first proposed by Davidson (1967). Later on the proposal was extended to quantifying over *states* and further enhanced by adding thematic relations in the representation to link the event with its participants (Parsons 1990). We will discuss the semantic relations in the next section. In section 3.2, we will present a *frame-based* representation in which the Davidsonian event variable corresponds roughly to a *situational* Index.

Type of Semantic Relations

We postulate two kinds of semantic relations: (a) relation between an eventuality and the participants involved in the eventuality. Such relations are called Thematic Relations and (b) Non-thematic relations such as tense, aspect, causal or temporal relations occurring between two eventualities and so on.

Thematic Relations

One or more participants can participate in the action or in a state-of-affair denoted by a verb. These participants occur in various thematic relation with respect to the verb. We have already come across three thematic relations in representing the sentence in (4). They are agent, theme and instrument. Agent is a volitional doer, Theme is the one being affected (or undergoing change of state) by the action and the Instrument is the apparatus (*sAdhana*) being used to accomplish the action. Here we list some other frequently occurring thematic relations.

Beneficiary: One who is benefited from the action

Source: Source of an action

Location: Place where the action takes place

Time: Time when the action takes place

Manner: Manner in which the action takes place

The relations are exemplified in the following small discourse. Since the prepositions and post-positions are the relation markers in English and Hindi respectively with respect to the verb, I am connecting them with the content words with an underscore and specifying the relations. The two verbs are underlined in the two sentences.

In English

Ram/**agent** went to_Delhi/**destination** from_Hyderabad/**source** by_train/**instrument** on_Tuesday/**time**. He/**agent** gave Meera/**beneficiary** an expensive gift/**theme** in_Delhi/**location**. In Hindi

□□□□/**agent** □□□□□□□□/**instrument** □□□□□□□□□□□□□□/**time**
 □□□□□□□□□□□□□□□□_□□/**source** □□□□□□□□/**destination** □□□□□ | □□□□□/**agent**

□□□□□_□□/location □□□□□_□□/beneficiary □□ □□□□□ □□□□□/theme
□□□□□ | - -

Non-thematic Relations

Apart from thematic relations, there can occur temporal or causal relations between more than one eventuality. For example, there are two events *Eating* and *Sleeping* which are temporally sequential in the following sentence:

(6) □□□□ □□□□ □□□ □□ □□ □□□□
 Ram roti eat cp
 sleep 3pt-masc
 “Ram slept after
 eating roti”

The formal representation of this sentence can be the following (ignoring tense):

We note in (7) that the agent information for *eating* and *sleeping* is explicitly stated in the representation.

The information of tense and aspect specified on verbs are required to be specified in the formal representation of the sentences. In a very simple way we can represent Past tense information as a time that Precedes the time of utterance and Future tense as a time that follows the utterance time. There can be interplay of tenses for complex sentences as is the case of the sentence in (6). We add tense information to the representation in (7) and present it below:

A complete tense logic is worked out in the seminal work of Reichenbach (1947). Here I postulate one predicate Cul which refers to *culmination of event*. The symbol '<' indicates the temporal sequence of the event *eating* and *sleeping*.

Sub-Event Analysis

An event may include subevents and the advantage of decompositional analysis of events can be illustrated with the following examples:

(10) Priya closed the door tight.

(11) □□□□ □□□□□ □□ □□□□□□□□□□□□□□□□□

Ram erg Hari acc loudly laugh-
 caus-3pt-masc “Ram made Hari
 laugh loudly”

In (9), the adjective 'tight' refers to the resultant state of the 'door' which comes into being as a result of the action *closing* performed by Priya. Thus the representation can be the following :

(11) $\exists ee'sxt(Is_a(e, Closing), agt(e, Priya), thm(e, x), Is_a(x, Door), Cul(e,t), Precedes(t, now), [Is_a(e', Being_closed), thm(e', x), Cause(e,e'), [Is_a(s, Closed), Holds(s), thm(s, x), result(e', s), Being_Tight(s)]])$

We notice that two subevents namely 'Cause to close' and 'Close' and one state 'In a state of being closed' have been postulated in the representation. The first event is the Cause of the second event. The resultant state is achieved as a result of 'Closing' and Tightness refers to the the resultant state in the representation.

Similarly, in (10), the adverb □□□□ □□'loudly' modifies the embedded event Laugh of which Hari is the agent. Thus the representation of (10) can be the following:

(12) $\exists ee'xt(Is_a(e, Making_laugh), Agt(e, Ram), Name(x, Hari), Thm(e, x) Cul(e,t), Precedes(t, now), [Is_a(e', Laughing), Agt(e', x), Cause(e,e'), Mod(e', Loud)])$

Exercise:

A. Identify the verb(s) in each sentence and mark thematic relation that the verb assigns to a noun

Example:

Mia left the house in order to learn music.

Leave: Mia (agent), house (theme)

Learn: Mia (experiencer), music (theme)

5. This mat weighs as much as that mat.
6. Smoking kills.
7. He drove the car too fast.
8. Mary fears thunder.
9. Sita made Meera feed her baby.
10. Ram dreamt a bad dream.
11. Ram sold his book to Mohan for Rs.100.
12. Ram bought a book to read to his students.
13. I lectured him because he needed it.
14. Amit is easy to please.

B. Give event semantics representation of the following sentences. Represent tense as well.

2. Amit closed the window
3. The builder will build one multi-storied apartment in six months at Nandigram.
4. Mother fed the child hot rice with daal.
5. Two girls are sitting in the park and chatting happily.
6. □□□□ □□□ □□□□ □□□□ □□□ □□□□ □□
7. □□□□□□□□□□ □□□ □□□□ □□□□□□□□□□ □□ □□□□□□□

Representing Meaning as Frames

The smallest parts carrying meaning are usually the lexical items. We present a *sign based* representation of lexical items (lexemes) and composite expressions (constructs). Saussurian concept of sign is modeled here as *typed feature structure*. A feature structure is a list of feature and value. Such feature structures can also be typed. Each type has appropriateness conditions expressing which features are appropriate for it. The value of a feature can be atomic or a typed feature structure. A sign is constituted of four features: PHON, FORM, SYN and SEM that constitute the description of any linguistic object. The value of PHON and FORM represent phonological and morphological information respectively. One can also add more features such as CONTEXT whose value will correspond to contextual information. In this work, we will mainly restrict ourselves to examining the correspondence between the value of SYN and SEM.

The values of AGT and THM within the FRAME ensures the index sharing between their value and the INDEX value of the two NPs that occur in the valence list (VAL) of the verb. This *re-entrant* structure ensures unification of the verb with its arguments and build the phrase structure (see section 4) . The next section introduces a *unification based approach* for constructing the syntax-semantic feature structure description of larger phrases and sentences.

Like signs, the *sem-objs* are also organized in a multiple inheritance hierarchy. For example, saying-fr is a subtype of statement-fr; eating- fr is a subtype of ingestion-fr. And since instances of a given subtype must specify values for all the features declared for that type and for those features inherited from its supertypes, each type of frame exhibits both idiosyncratic properties and the properties of more general frame classes. The frame hierarchy is used to account for many lexical entailments.

Exercise

1. Make a multiple inheritance hierarchy for noun-lxm, common-noun-lxm, proper-noun-lxm. Mention typed feature structure for each lexeme.

2. What is *typed features structure*? Explain with examples.

Constructing Semantic Structure

How do we automate the process of assigning semantic representations to sentences of human language? The common strategy of producing meaning representation for natural language expressions underlies a *compositional* analysis. This strategy assumes that the meaning of a composite expression is computed from the meanings of its parts. When compatible lexical items are combined, we get the description of composite constructions finally leading to the composition of sentences. These composite constructions fulfill certain well-formedness principles which ensure the grammaticality of larger expressions and finally that of sentences. The two dominant methods of composition are the following: (a) one is based on Unification based approach and (b) the other one is based on lambda calculus. We will examine in this paper a unification based approach of composition.

UNIFICATION AS A WELL-FORMEDNESS CONSTRAINT CHECKING OPERATION

Unification is an operation that amalgamate compatible partial information and result in wellformed *composite syntactic construction* of larger phrases. We will now examine the syntax-semantics composition of a very simple sentence *ravi dORegA* “Ravi will run” and the same sentence with a modifier as in *ravi teza dORegA* “Ravi will run fast” later. We will also examine two types of modifiers of verbs and nouns, namely intersective and scopal modifier. Depending on whether they modify a noun or a verb, they are categorized as adjective and adverb respectively.

Reference and Compositionality

If the direct **reference** theory is true and **compositionality** is true, it follows that two sentences that differ only in the substitution of one co-referring name for another will mean the same thing. ... Whenever (a) is true, (b) is also true, and vice versa.

Compositionality: the interpretations of phrases have to give us correct truth-conditions when the phrases are contained in declarative sentences I “Picasso”: pp I “is alive”: $\{x: x \text{ is alive}\}$ I “Picasso is alive”: $pp \in \{x: x \text{ is alive}\}$

Sentences may have the same true conditions but distinct meanings. I Sets $\{x: x \text{ is alive}\}$ and $\{x: x \text{ is alive and } x \text{ is not a rock}\}$ are identical I Identical meaning? I Picasso is alive. I Picasso is alive and is not a rock.

First-Order Logic in Artificial intelligence

In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.

- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

First-Order logic:

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the

objects in a more easy way and can also express the relationship between those objects.

- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
 - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus,
 - **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between
 - **Function:** Father of, best friend, third inning of, end of,
- As a natural language, first-order logic also has two main parts:
 - a. **Syntax**
 - b. **Semantics**

Syntax of First-Order logic:

The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL.

Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf,
Connectives	$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$

Equality	$==$
Quantifier	\forall, \exists

Atomic sentences:

- o Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- o We can represent atomic sentences as **Predicate (term1, term2,, term n)**.

Example: Ravi and Ajay are brothers: \Rightarrow Brothers(Ravi, Ajay).
Chinky is a cat: \Rightarrow cat (Chinky).

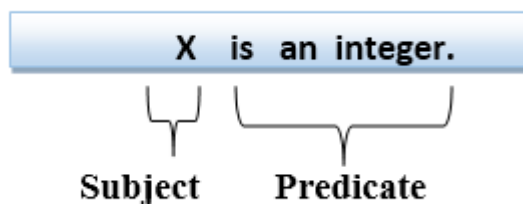
Complex Sentences:

- o Complex sentences are made by combining atomic sentences using connectives.

First-order logic statements can be divided into two parts:

- o **Subject:** Subject is the main part of the statement.
- o **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
 - a. **Universal Quantifier, (for all, everyone, everything)**
 - b. **Existential quantifier, (for some, at least one).**

Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol \forall , which resembles an inverted A.

UNIT 4

Computational Lexicography

Computational lexicology

Covers aspects of computer use in lexicological research, and focuses on the computational representation of lexicon, on computational carried out on lexical data, and on relationship between computational lexicons on the one hand and other component of the various NLP system on the other.

Computational Lexicography

Covers computational methods a tools designed to assist the various lexicographical tasks, including the prepared lexicographical evidence from many sources the recording the database form of the relevant linguistic inform the editing of lexicographic entities, and the dissemination of lexicographical products.

Lexicography

The collection of lexical items and description of the way they are used. Typical lexicographic products include not only print dictionaries of various types, but also concordance, indexes, terminologies and so on. Language can be studied from the point of view of language structure and language use. Study of language structure is called structural or formal linguistic study. Study of language use is called as functional linguistic study. Languages are described qualitatively in terms of grammatical units like nouns, verbs, noun phrases, verb phrases, subject, object, agent, goal, etc. to explain the structure of the language. Most of the grammars take a sentence as the minimum unit for the description of the structure. The structure has been studied from different view points and many linguistic theories have emerged to account the syntactic pattern of the sentence

Collect data

When planning on how best to collect data in Step 4, it is important to be aware of the practical considerations and best practices for addressing logistical challenges organizations often face at this stage of the process. Implementing a data collection plan requires attention to matters such as:

- Getting buy-in from senior leadership and key stakeholders, in or outside of the organization. This group could include boards of directors, management committees, union representatives, employees, community groups, tenants, customers and service users.
- Establishing a steering committee or selecting a person(s) to be consulted and held accountable for all major decisions about the data collection process, such as design, logistics, communication management, coordination and finances.
- Determining who will collect the data (e.g., experts or trained employees).
- Identifying the logistics, resources, technology and people needed to develop and implement a data collection initiative.
- Anticipating and addressing key stakeholder concerns and questions about the project.

- Designing a communication and consultation strategy that will explain the data collection initiative and encourage the highest possible participation rate.
- Protecting privacy and personal information by using carefully controlled procedures for collecting, storing and accessing data that comply with privacy, human rights and other legislation. Dignity and confidentiality must be respected.
- Minimizing the impact and inconvenience for the people affected in the workplace or service environment, which includes choosing the best time to collect the data.
- Aiming for flexibility to allow for changes without great expense or inconvenience.
- Considering a test period or a pilot phase to allow you to improve and modify data collection methods, as may be needed.

Step 5: Analyze and interpret data

Step 5 involves analyzing and interpreting the data collected. Whether quantitative and/or qualitative methods of gathering data are used, the analysis can be complex, or less so, depending on the methods used and the amount of data collected.

Explaining the technical steps involved in analyzing and interpreting data is beyond the scope of this guide. An organization will have to determine whether it has the internal capacity and expertise to analyze and interpret data itself, or whether it will need the help of an external consultant.

A smaller organization that has basic data collection needs may be able to rely on internal expertise and existing resources to interpret the meaning of gathered data.

Example: An organization with 50 employees wants to find out if it has enough women working in management positions, and if there are barriers to equal opportunity and advancement. The organization counts the number of female employees it has (25), and determines how many of these employees are working in supervisory and management positions (two). A few motivated employees identify some issues of concern, like gender discrimination, that may have broader implications for the organization as a whole.

After deciding to do an internal and external assessment (Step 1), and gather qualitative data using focus groups and interviews with current and past employees,

senior leadership decides that barriers exist for women in the organization's recruitment, hiring, promotion and human resources policies, processes and practices. Efforts are made to work with female employees, human resources and other staff to address these barriers. The organization makes a commitment to foster a more equitable, inclusive work environment for all employees.

Step 6: Act on results

Once an organization has analyzed and interpreted the results of the data collected, it may decide to act on the data, collect more of the same type of data or modify its approach.

Quantitative and qualitative information can provide a solid basis for creating an effective action plan designed to achieve strategic organizational human resources, human rights, equity and diversity goals identified through the data collection process. If an organization feels it has enough information to develop an action plan, it should consider including the following elements:

- a summary of the results of the analysis and interpretation of the data
- identification of the barriers, gaps and opportunities that exist or may exist for Code-protected persons and other individuals/groups based on non-Code grounds
- steps that will be taken to address these barriers, gaps or opportunities now and in the future
- realistic, attainable goals with short-term and longer-term timelines
- input sought from stakeholders and affected communities
- how progress in meeting these goals will be monitored, evaluated and reported.

In some cases, an organization may decide that it needs to collect more information because there are gaps in the data collected, or areas where the data is unclear or inconclusive. This may prompt them to conduct a more detailed internal and external assessment (go back to Step 1) or try another approach.

In the end, there is no one or "right way" to conduct a data collection initiative. The experiences of Mount Sinai Hospital, KPMG Canada, the Keewatin-Patricia District School Board, TD Bank Financial Group, the University of Guelph and the *DiverseCity Counts* project and featured in the Appendices reflect this statement, yet also show some similarities in terms of the best practices and lessons learned.

Six steps to success

Step 1: Identify issues and/or opportunities for collecting data

Step 2: Select issue(s) and/or opportunity(ies) and set goals

Step 3: Plan an approach and methods

- Who will the data be collected about?
- Who will the group of interest be compared to?
- What locations or geographical areas will the data be gathered from?
- What categories will be used to identify the group of interest and comparator group?

How should data be collected?

- Qualitative Data
- Quantitative Data

What sources of data should be used to collect information?

- Pre-existing or official data
- Survey data
- Interviews and focus groups
- Observed data
 - Identify issues and/or opportunities for **collecting data**. ...
 - Step 2: Select issue(s) and/or opportunity(ies) and set goals. ...
 - Step 3: Plan an approach and methods. ...
 - Step 4: **Collect data**.

Data entry

Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs. The simplest kind of data entry consists merely of pointing at something -- selecting an item or designating a position on a computer-generated display. In more complicated modes of data entry, a user may have to control the format of data inputs as well as their contents. Thus questions of format control in text entry/editing and graphic interaction may properly be considered questions of data entry.

Note, however, that user inputs which initiate or interrupt transactions -- such as command entries, or control entries selected from a displayed menu or by function keys - - pose rather different questions of design. Such control entries are discussed in Section 3 of these guidelines.

Data can be entered into a computer in a variety of different ways. Users might designate position or direction by pointing at a display. Users might enter numbers, letters, or more extended textual material by keyed inputs, or in some applications by spoken inputs. Data might be keyed into displayed forms or tables, into constrained message formats, or as free text. In graphic interaction users might draw pictures or manipulate displayed graphic elements. These different types of data entry all merit consideration here.

The computer will also play a role in the data entry process, guiding users who need help, checking data entries to detect errors, and providing other kinds of data processing aids. A designer of user interface software must be concerned about computer processing logic as well as data input by the user.

Data entry is heavily emphasized in clerical jobs, and many other jobs involve data entry to some degree. Because data entry is so common, and because inefficiencies caused by poorly designed data entry transactions are so apparent, many published recommendations for good user interface design deal with data entry questions. Human factors specialists can probably give better advice about data entry than about any other functional area of user interface design.

Data entry requires hardware, and the proper design of input devices has received considerable attention, including concern for standardization of keyboard layouts. Future advances in hardware design may well influence data entry tasks, as suggested by current advocacy of voice input.

But the major need in today's information systems is for improving the logic of data entry, and it is there that design guidance should prove most helpful. Thus the guidelines presented here deal with data entry functions, insofar as possible, without regard to their hardware implementation.

The general objectives of designing data entry functions are to establish consistency of data entry transactions, minimize input actions and memory load on the user, ensure compatibility of data entry with data display, and provide flexibility of user control of data entry. Stated in such general terms, these principles do not provide helpful guidance to designers. Somehow these general ideas must be converted into more specific guidelines.

The process of converting general principles into more detailed guidelines will lead to a considerable proliferation of ideas. With regard to minimizing input actions, one guideline might be that a user should not have to enter the same data twice. Probably every designer knows that, even if it is sometimes forgotten. A related guideline might be that a user should not have to enter data already entered by another user. That seems to make good sense, although one could imagine occasional exceptions when cross validation of data inputs is required.

How can duplicative data entry be avoided in practice? The solution lies in designing the user interface (programming the computer) to maintain context. Thus when a user identifies a data category of interest, say a squadron of aircraft, the computer should be able to access all previously entered data relevant to that squadron and not require the user to enter such data again.

In repetitive data entry transactions the user should have some means of establishing context. One method is to allow users to define default entries for selected data items, in effect telling the computer that those items will stay the same until the default value is changed or removed. If a user enters one item of data about a particular squadron, it should be possible to enter other items thereafter without having to re-identify that squadron.

Context should also be preserved to speed correction of input errors. One significant advantage of on-line data entry is the opportunity for immediate computer validation of user inputs, with timely feedback so that a user can correct errors while the data are still fresh in mind and while documented source data are still at hand. Here the computer should preserve the context of each data entry transaction, saving correct items so that the user does not have to enter those again while changing incorrect items.

Preservation of context is, of course, important in all aspects of user-system interaction, with implications for data display, sequence control and user guidance, as well as for data entry. The importance of context is emphasized again in the discussion of those other functional areas. Another important design concept is flexibility. It is easy to say that the interface should adapt flexibly to user needs, but the specific means of achieving such flexibility must be spelled out in design guidelines. For data entry functions it is important that the pacing of inputs be controlled flexibly by the user. Tasks where the pacing of user inputs is set by a machine are stressful and error-prone.

Aside from flexibility in pacing, users will often benefit from having some flexible choice in the ordering of inputs. What is needed for interface design is some sort of suspense file to permit flexible ordering of data entries, including temporary omission of unknown items, backup to correct mistaken entries, cancellation of incomplete transactions, etc.

As noted above, users may also benefit from flexibility in defining default options to simplify data entry during a sequence of transactions. Some systems include only those defaults anticipated by the designers, which may not prove helpful to the user in a particular instance. Thus the concept of flexibility is related to maintaining context, and is related also to many other aspects of interface design.

The guidelines proposed here deal with data entry in terms of specific functions, covering different kinds of data entry and different kinds of computer processing support. Some topics, such as "abbreviation", which pertain to all data entry are covered in an initial group of guidelines dealing generally with the subject. A summary of the functional coverage in this section is presented on the next page. These guidelines recommend specific ways to accomplish the fundamental design objectives for data entry.

Objectives

Consistency of data entry transactions Minimal entry actions by user Minimal memory load on user Compatibility of data entry with data display Flexibility for user control of data entry

1.0 General

Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs.

1.0/1 Data Entered Only Once

Ensure that a user need enter any particular data only once, and that the computer can access those data if needed thereafter for the same task or for different tasks.

Comment

In effect, this recommendation urges integrated and flexible software design so that different programs can access previously entered data as needed. Requiring re-entry of data would impose duplicative effort on users and increase the possibility of entry errors.

See also

1.8/9

1.0/2 Entry via Primary Display

When data entry is a significant part of a user's task, entered data should appear on the user's primary display.

Example

As a negative example, entry via typewriter is acceptable only if the typewriter itself, under computer control, is the primary display medium.

Comment

When the primary display is basically formatted for other purposes, such as a graphic display for process control, a separate window on the display may have to be reserved for data entry.

1.0/3 Feedback During Data Entry

Provide displayed feedback for all user actions during data entry; display keyed entries stroke by stroke.

Exception

For reasons of data protection, it may not be desirable to display passwords and other secure entries.

1.0/4 + Fast Response

Ensure that the computer will acknowledge data entry actions rapidly, so that users are not slowed or paced by delays in computer response; for normal operation, delays in displayed feedback should not exceed 0.2 seconds.

Example

A key press should be followed by seemingly immediate display of its associated symbol, or by some other appropriate display change.

Comment

This recommendation is intended to ensure efficient operation in routine, repetitive data entry tasks. Longer delays may be tolerable in special circumstances, perhaps to reduce variability in computer response, or perhaps in cases where data entry comprises a relatively small portion of the user's task.

1.0/5 Single Method for Entering Data

Design the data entry transactions and associated displays so that a user can stay with one method of entry, and not have to shift to another.

Example

Minimize shifts from lightpen to keyboard entry and then back again.

Example

As a negative example, a user should not have to shift from one keyboard to another, or move from one work station to another, to accomplish different data entry tasks.

Comment

This, like other guidelines here, assumes a task-oriented user, busy or even overloaded, who needs efficiency of data entry.

1.0/6 Defined Display Areas for Data Entry

Where data entry on an electronic display is permitted only in certain areas, as in form filling, provide clear visual definition of the entry fields.

Example

Data entry fields might be underlined, or perhaps highlighted by reverse video.

Exception

For general text entry of variable (unrestricted) length, no field delimiters are needed. In effect, keyed text entries can replace nothing (null characters).

Comment

Display formats with field delimiters provide explicit user guidance as to the location and extent of data entry fields. Where delimiters extend throughout an entry field, as in underlining, then any keyed data entries should replace the delimiter characters on the display.

Dictionary types:

1. monolingual dictionary (thesaurus, conceptual)
2. special dictionary (slang, etymology)
3. encyclopedic (real world reference)
4. translation dictionary (only for Equivalents)
5. corpus (concordance, KWIC)
6. vocabulary (domain term inventory)

Objectives of computational Lexicography

1. To investigate the design, construction and use of electronic dictionaries in natural language processing.
2. the notion of re-usability of lexical resources is taken as a focus, enabling considerations of existing lexical resources, including publishers (MRDs) and lexical database, as well as the constructions of new resources.

The computational lexicography involves the following processes:

1. using computers to assist ID, capture, encoding, dissemination of lexicographic information.
2. Implementing/ leveraging/ integrating lexicographic resources in computational tasks.
3. supplying/ comparing/ evaluating MRDs.
4. tools for all of above.

Main sources of information for CL work.

- i. Machine Readable Dictionary (MRD)
- ii. Text corpora

Machine Readable Dictionary

- Machine-readable dictionary (MRD) is a dictionary stored as machine (computer) data instead of being printed on paper.
- It is an electronic dictionary and lexical database.
- A machine-readable dictionary is a dictionary in an electronic form that can be loaded in a database and can be queried via application software.
- It may be a single language explanatory dictionary or a multi-language dictionary to support translations between two or more languages or a combination of both.
- Translation software between multiple languages usually apply bidirectional dictionaries.
- An MRD may be a dictionary with a proprietary structure that is queried by dedicated software (for example online via internet) or it can be a dictionary that has an open structure and is available for loading in computer databases and thus can be used via various software applications.
- Conventional dictionaries contain a lemma with various descriptions.
- A machine-readable dictionary may have additional capabilities and is therefore sometimes called a smart dictionary.
- An example of a smart dictionary is the Open Source Gellish English Dictionary.
- The term dictionary is also used to refer to an electronic vocabulary or lexicon as used for example spelling checkers.
- If dictionaries are arranged in a subtype-supertype hierarchy of concepts (or terms) then it is called a taxonomy.
- If it also contains other relations between the concepts, then it is called an ontology.
- Search engines may use either a vocabulary, a taxonomy or an ontology to optimise the search results.
- Specialised electronic dictionaries are morphological dictionaries syntactic dictionaries.
- TYPES OF DICTIONARY:
- Bilingual Dictionary

- Monolingual Dictionary
 - Etymological Dictionary
 - Crossword Dictionary
 - Rhyming Dictionary
 - Mini-Dictionary
 - Pocket Dictionary
 - Thesaurus
 - Glossary
-
- A *bilingual dictionary* gives words in two languages. Each language is grouped alphabetically in separate halves of the book, with translations into the other language.
 - A *monolingual dictionary* uses the same language for the words and their definitions.
 - An etymological dictionary traces a word's development over time, giving historical examples to show changes.
 - A crossword dictionary has words grouped together by the number of letters in the word to help people find words of a certain length to complete their crossword puzzles.
 - A Rhyming Dictionary is one where words are grouped together by their end sounds. When two words end with the same sound, they rhyme, used more frequently in poetry than prose as an effect.
 - A *mini-dictionary* is a little dictionary, also called a pocket dictionary.
 - A pocket dictionary is a small portable dictionary designed to be carried around. Consequently, they often have tough covers to withstand the perils of travelling.
 - A thesaurus is a book that organises words by categories and concepts, so synonyms and near-synonyms will be grouped together.
 - A glossary is a list of words or phrases used in a particular field with their definitions. Glossaries are often found at the back of a specialist or academic book as an appendix to the text.

- Machine translation is the task of automatically converting source text in one language to text in another language.
- In a machine translation task, the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language.

The role of Corpus Linguistics in Lexicography

- Languages can also be studied quantitatively in terms of frequency, place of occurrence, pattern of occurrence, etc. of various linguistic units in a text. The quantitative study basically needs a large quantum data and a mechanism to browse them fast. Now the computer technology facilitates to store and study a huge texts to the tune of hundreds of millions of words in fewseconds or minutes. A new method of language study called corpus linguistics has emerged in recent years.
- Corpus is a large collection of written or spoken texts available in machine readable form accumulated in scientific way to represent a particular variety or use of a language. It serves as an authentic data for linguistic and other related studies. The size, text type, organization, accessing method, etc. are some of the basic features of a corpus which have to be carefully decided while generating a corpus. There are different types, which are again.

The role of corpus linguistics in lexicography. Corpus linguistics is a study of language and a method of linguistic analysis which uses a collection of natural or “real word” texts known as corpus. Corpus linguistics is used to analyse and research a number of linguistic questions and offers a unique insight into the dynamic of language which has made it one of the most widely used linguistic methodologies.

Since corpus linguistics involves the use of large corpora that consist of millions or sometimes even billion words, it relies heavily on the use of computers to determine what rules govern the language and what patters (grammatical or lexical for instance)

occur. Thus it is not surprising that corpus linguistics emerged in its modern form only after the computer revolution in the 1980s. The Brown Corpus, the first modern and electronically readable corpus, however, was created by Henry Kucera and W. Nelson Francis as early as the 1960s.

MRD

The interaction between computational Linguistics and lexicographers has in the past been virtually limited to building sample tools for producing indexes and concordance of large corpora.

1. lexicographers should use the techniques developed by computational linguistics for browsing machine readable dictionary in order to reuse their data.
2. in future electronic publishing benefit end user by improving access to the wealth of information currently available in dictionaries but not easily retrievable via alphabetical order alone.

The processing of text corpora

- one of the reason is that computational linguistics which covers both linguistic with and linguistics for computers, is developing very fast so that what held true today may be superseded tomorrow.
- Another obvious reason is that computational linguistics is so vast a domain that so single individual can not be expected to cover the whole of it, nor can the combined experience of three workers in the field remedy this fully.

What do we mean by lexicographical informations?

A text could be defined as a set of one or more (written or spoken) words forming a consistent whole from a semantic, syntactic, and pragmatic point of view.

Lexicographer make use of 'Primary' texts and of 'secondary' texts.

Texts---→ Processing ---→ Lexicographical information

By ‘lexicographic information’ we mean the kind of information one can expect to find in a dictionary.

- Gives an access to naturalistic linguistic information. As mentioned before, corpora consist of “real word” texts which are mostly a product of real life situations. This makes corpora a valuable research source for dialectology, sociolinguistics and stylistics.
- Facilitates linguistic research. Electronically readable corpora have dramatically reduced the time needed to find particular words or phrases. A research that would take days or even years to complete manually can be done in a matter of seconds with the highest degree of accuracy.
- Enables the study of wider patterns and collocation of words. Before the advent of computers, corpus linguistics was studying only single words and their frequency. Modern technology allowed the study of wider patterns and collocation of words.
- Allows analysis of multiple parameters at the same time. Various corpus linguistics software programmes, [online marketing](#) and analytical tools allow the researchers to analyse a larger number of parameters simultaneously. In addition, many corpora are enriched with various linguistic information such as annotation.
- Facilitates the study of the second language with the use of natural language allows the students to get a better “feeling” for the language and learn the language like it is used in real rather than “invented” situations.

Corpus Linguistics Software Functions

Corpus linguistics software allows the user to research and study many linguistic questions. Functions included vary greatly from one type of software to another but some basic functions such as the ability to search for particular words or phrase, their sequence, frequency and concordance (the immediate context of the word or phrase) are found on most software programmes. Typically included are also the ability of sorting (for example listing the words in alphabetical order) and “thinning” or remove irrelevant results. More sophisticated programmes, of course, come with a larger number of functions which provide more detailed information on linguistic questions.

Corpus linguistics is used for a variety of linguistic research but it also has a number of practical applications. The study of language with the use of “real word” texts or corpora is most often used in:

- Lexicography. Corpus linguistics plays an important role in compiling, writing and revising dictionaries as within a few seconds, the linguist can get examples of words or phrases from millions of spoken and written texts. And since corpora continue to grow and are constantly being expanded with new texts, lexicographers have an instant access to up-to-date information.
- Corpus Linguistics Terms and Their Meanings
- Corpus (plural corpora). It refers to a collection of systematically or randomly collected texts of natural language which is electronically stored and processed. Corpus can consist of texts in a single or multiple languages. It contains a large number of texts which allow the researchers to analyse linguistic rules but the corpus does not represent the entire language, no matter how large it is.
- Multilingual corpus. Like its name suggests, multilingual corpus consists of texts in multiple languages.
- Parsed corpus (treebank). It is a collection of texts in naturally occurring language in which each sentence is parsed - syntactically analysed and annotated. Syntactic analysis is typically given in a tree-like structure which is why parsed corpus is also known as treebank.
- Parallel corpora. The term refers to a collection of texts which are translations of each other.
- Annotation. It refers to an extension of the text by addition of various linguistic information. Examples include parsing, tagging, etc. Annotation is often used in reference to corpora as opposed to annotated corpora which consist of plain text in the raw state.
- Collocation. It refers to a sequence or pattern in which the words appear together or co-occur.
- Concordance. The term encompasses a word or phrase and its immediate context. In corpus linguistics, concordance is used to analyse different use of a single word, word frequency and phrases or idioms.

- Orthography. It is a standardised writing system of a particular language and includes various grammatical rules such as spelling, capitalisation and punctuation marks. Orthography can pose a problem in analysis of writing systems which use accents because the native speakers of these languages sometimes use alternative characters to the accented letters or omit them completely.
- Token. It is an occurrence of an individual word which plays an important role in the so-called tokenisation that involves division of the text or collection of words into tokens. This method is often used in the study of languages which do not delimit words with space.
- Lemmatisation. The term derives from the word lemma which refers to a set of different forms of a single word such as laugh and laughed for example. Lemmatisation is the process of grouping of the words that have the same meaning.
- Wildcard. It refers to special characters such as question mark (?) or asterisk (*) which can represent a character or word.
- 3A perspective. It is a research method that is used in corpus linguistics which was introduced by S. Wallis and G. Nelson. 3A stands for annotation, abstraction and analysis.

What Corpus Linguistics Does Not

- Does not explain why. The study of corpora tells us what and how happened but it does not tell us why the frequency of a particular word has increased over time for instance.
- Does not represent the entire language. Corpus linguistics studies the language by using randomly or systematically selected corpora. They typically consist of a large number of naturally occurring texts, however, they do not represent the entire language. Linguistic analyses that use the methods and tools of corpus linguistics thus do not represent the entire language.

Notable Corpora

Corpus, a large collection of texts in naturally occurring language is the very basis of corpus linguistics. It is used for linguistic analyses but it may also be used as a tool in

second language teaching and learning. A corpus may consist of texts in a single or multiple languages and may encompass written or spoken language, or both.

Some of the most notable corpora include:

- Brown Corpus (the Brown Standard Corpus of Present-Day American English). The corpus which consists of about 500 English language texts that total about 1 million words was compiled by Henry Kucera and W. Nelson Francis in the 1960s. It is small in comparison to modern corpora but it is almost always found on the lists of notable corpora because it is the first modern and electronically readable corpus.
- British National Corpus. It consists of a wide range of written and spoken texts in English language, totalling 100 million words. It was created by three publishers - the Oxford University Press, W. & R. Chambers and Longman, two universities - the Oxford University and Lancaster University and the British Library between 1991 and 1994. However, the original British National Corpus was slightly revised in the second (2001) and third (2007) editions.
- Oxford English Corpus. It is a huge corpus of English language totalling over 2 billion words. The texts included in the corpus are taken from all sorts of sources, ranging from literary works to the language in forums and chatrooms. The Oxford English Corpus is used by the Oxford University Press' linguistic research department and the creators of the Oxford English Dictionary.
- American National Corpus. It is the American English equivalent to the British National Corpus, however, in contrary to the latter that contains 100 million words, the American National Corpus currently contains about 22 million words of American English spoken and written texts. It distinguishes itself from other English language corpora by being richly annotated. The American National Corpus is being developed since 1990.
- International Corpus of English. It consists of a set of corpora which contain variations of English language from countries where English is either the first or official second language. Each set of the International Corpus of English contains 1 million word texts that have been created after the year 1989. The project is ongoing since 1990.

- Scottish Corpus of Texts and Speech. The collection of written and spoken texts in Scottish English and Scots after 1940 is available online for free since 2004. In 2007, the corpus reached a total of 4 million words. The project was carried out by the School of Critical Studies at Glasgow University.
- History of Corpus Linguistics
- History of corpus linguistics is typically divided into two periods: - early corpus linguistics, also known as pre-Chomsky corpus linguistics and - modern corpus linguistics
- The early examples of corpus linguistics date to the late 19th century Germany. In 1897, German linguist J. Kading used a large corpus consisting of about 11 million words to analyse distribution of the letters and their sequences in German language. The impressively sized corpus that corresponds with the size of a modern corpus was revolutionary at the time.
- Other early linguists to use corpus to study language include Franz Boas (Handbook of Native American Indian Languages, 1911), Zellig Harris (Methods in Structural Linguistics, 1951), Charles C. Fries (The structure of English, 1952), Leonard Bloomfield (Language, 1933), Archibald A. Hill and others, mostly American structural and field linguists. Some of them such as Fries and A. Aileen Traver also started to use corpus in pedagogical study of foreign language.
- In 1961, Henry Kucera and W. Nelson Francis from the Brown University started to work on the Brown University Standard Corpus of Present-Day American English, commonly known simply as the Brown Corpus which is the first modern, electronically readable corpus. It consists of 1 million word American English texts that are organised into 15 categories. For the modern standards of corpus linguistics, the Brown Corpus is kind of small, however, it is widely considered one of the most important works in history of corpus linguistics. But this was also the time of Chomsky's criticism of corpus linguistics which would result in a period of decline.
- Chomsky rejected the use of corpus as a tool for linguistic studies, arguing that linguist must model language on competence instead of performance. And according to Chomsky, corpus does allow language modelling on competence.

Corpus linguistics was not abandoned completely, however, it was not until the 1980s when linguists began to show an increased interest in the use of corpus for research.

- The revival of corpus linguistics and its emergence in the modern form was greatly influenced by the advent of computers and network technology in the 1980s which allowed the linguists to use electronic language samples as well as electronic tools. The use of computers, however, dates back to the early 1970s when the Montreal French Project developed the first computerised form of spoken language, while Jan Svartvik began to work on the London-Lund corpus with the aid of the Brown Corpus and the Survey of English Usage (SEU) at University College London.
- All mentioned works before the 1980s as well as the early examples of corpus linguistics paved the way to modern study of language on the basis of corpora as we know it today. The term corpus linguistics has been finally adopted after J. Aarts and W. Meijs published *Corpus linguistics: Recent developments in the use of computer corpora in English language research* in 1984.

Corpus Linguistics Software

Today, corpus is often used as a synonym for electronic corpus. It is the advent of technology, in the first place of computers that stimulated the revival of corpus linguistics and remains the driving force of the study of language on the samples of “real word” texts. Thanks to various corpus linguistics software programmes, the same analysis that took years for the German linguist J. Kading to complete manually in the late 19th century takes only a few moments with the computer.

Corpus linguistics software is a computer programme which is the very most important tool for every linguist who studies language on the basis of corpora. Like its name suggests, it is designed specially for the needs of corpus linguistics. Its main advantage is of course the speed of processing as it turns corpora into databases which can be searched, similarly as browsing the web. But it also provides the linguists with a variety of tools that allow analysis of multiple linguistic questions simultaneously. Lastly, it eliminates the risk of human error and as a result, modern corpus linguistics can

analyse much larger corpora than before the advent of computers with the highest degree of accuracy.

Unit–V

Application of Computational Linguistics

Machine Translation

Information plays a vital role in the growth of science and technology. Similarly the growth and development of a language, its culture and literature depends very much on sharing or spreading information. Hence, acquiring, transferring, disseminating and diffusing information is crucial to the development of a human society. Many of the decision-making processes and the research programs are based on the most valuable information concerning the respective areas. Translation is one of the strategies that plays an important role in disseminating information contents from one language to another. Since the last century this role has been given to Machine Translation.

Machine translation, sometimes referred to by the abbreviation MT, is a sub-field of Computational Linguistics that investigates the use of computer software to translate text or speech from one natural language to another. At its basic level, MT performs simple substitution of words in one natural language for words in another. Using corpus techniques, more complex translations maybe attempted, allowing for better handling of differences in linguistic typology, phrase recognition, and translation of idioms, as well as the isolation of anomalies.

A machine translation can be classified into several types:

Human-Assisted Machine Translation (HAMT) is a style of translation, in which computer system does most of translation, appealing in case of difficulty to a human for help. It involves the help of a human to pre-edit the source language text before being fed to the computer for translating into a target language. Due to the information explosion and the necessity of faster and cheaper translation, HAMT is very suitable for translating texts in science and technology, weather reports, equipment manuals, military records,

laboratory procedures, articles, research monographs, etc. Since human intervention is required in some form either pre-editing a SL text or feeding the input or specifying certain grammatical rules, most of the currently available MT systems belong to HAMT.

Machine-Assisted Human Translation (MAHT) is a style of translation, in which a human does most of the work, but uses one or more computer systems, mainly as resources such as dictionaries and spelling checkers, as translation tools. The translation is aided by the computer, providing suitable word equivalents from the already stored glossary of words. In certain technical and scientific area conveying the basic meaning is considered to be sufficient through word-to-word translation by MAHT.

In recent times, the HAMT and MAT are grouped together under one term *Computer-Aided Translation (CAT)*, which cover both of the meanings.

Fully Automated Machine Translation (FAMT) is performed without the intervention of a human being during the translation process. Such a completely automated translation perhaps will become a reality one day when the entire domain knowledge with complete and appropriate translation rules are modeled to resolve the syntactic and semantic ambiguities of a natural language (NL). The extensive research in the area of Natural Language Processing (NLP) and the various tools of Artificial Intelligence (AI) would certainly lead the way to achieve this goal in the future at least in the field of science and technology.

To know why we need MT, we have to look closer at functions of machine translation. There are several goals MT is used for:

Dissemination: the production of translations of ‘publishable’ quality; not necessarily texts that are actually published but texts that are of that quality. Such texts are required usually by organizations and usually involve professional translators. The ‘raw’ untreated output from MT systems is inadequate, and publishable quality means human assistance; e.g. post-editing (revision) of the output text, pre-editing of the input, using a controlled

language, or restricting the system to a specific subject domain. In general it has been found (through the experience of using MT) that the more the subject domain of an application can be restricted the more successful the system is, in terms of quality.

Assimilation: the translation of texts for monitoring (or ‘filtering’) or skimming information, or the translation of texts for occasional users (e.g. non-specialist general public), where the ‘raw’ output from the system doesn’t need to be edited; in other words, where recipients can accept poor quality as long as they can get an idea of what the text conveys. (In fact, we find that many MT systems are useful only in this function – their outputs are unsatisfactory as drafts for dissemination tasks.)

Interchange: the communication between different languages by individuals, by correspondence, by email or by telephone. Here again the quality of the translation (and/or closeness to the original) is not so important, as long as people get the information they want, understand the message they receive, or manage to convey their intentions.

Database access: the use of translation to assist in getting information from a database in a foreign language, one that the user does not well understand – i.e., these days this means mainly the use of translation aids for searching the Internet, for accessing web pages. While dissemination and assimilation have been traditional uses of MT and translation tools from the beginning, their uses for interchange and databases access are recent applications which are growing rapidly, primarily with the growth of the Internet itself.

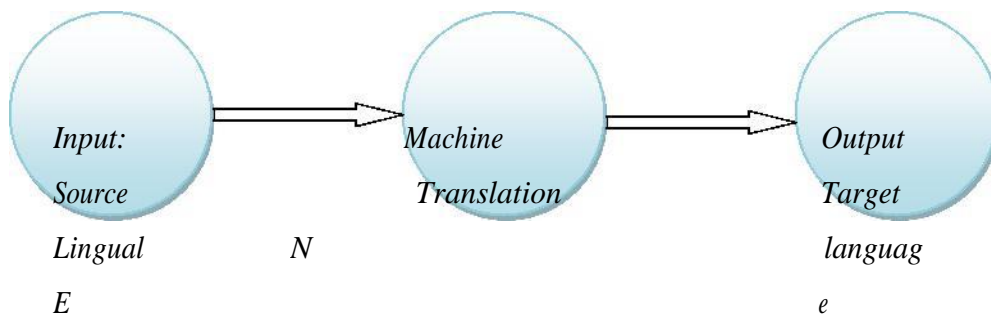
Architectures and Paradigms of Machine translation Systems

The success of a MT partly depends on inner architecture of a system – the process of handling language in synthesis and generation to give final output. Systems can be built by different axis: architectures and paradigms. Architectures refer to the actual processing design (i.e., direct, transfer, intralingual), whereas the paradigm refers to

informational components that aid the processing design (knowledge-based, example-based, statistics-based, etc.). Let us see what type of architectures and paradigms of MT systems have been developed by computational linguists.

In general, an MT system involves feeding the program with a text in source language (SL), the text is undergoing translation process and, then, the machine gives out output in the target language. It looks like the following (Graph 1):

Graph 1.



2.1. Architectures

The earliest historically is the “direct approach”, adopted by most MT systems of what has come to be known as the first generation of MT systems. In response to the apparent failure of this direct architecture strategy, two types of “indirect architecture” were developed: the “transfer”, and the use of an “interlingua” approaches. Systems of the latter architectures are sometimes referred to as second generation systems.

Direct architectures

Direct MT is the one of the simplest machine translation architectures. In direct MT a direct word byword translation of the input source is carried out with the help of a bilingual dictionary and after which some syntactical rearrangement is made the **direct method** lacks any kinds of intermediate stages in translation processes, where the processing of the SL input text leads “directly” to the desired TL output text. There is no analysis of syntactic structure or of semantic relationships. In other words, lexical identification depends on morphological analysis and leads directly to bilingual dictionary look-up providing TL word equivalences. Some local reordering rules follows to give more acceptable TL output, perhaps moving some adjectives verb particles, and then the TL text is produced. This approach is unidirectional and only takes one language pair into consideration at a time.

The severe limitations of this approach are obvious. It can be characterized as “word-for-word” translation with some local word-order adjustment. It gives the kind of translation quality that might be expected from someone with a very cheap bilingual dictionary and only the most basic knowledge of the grammar of the target language: frequent mistranslations at the lexical level and largely inappropriate syntax structures which mirrored too closely those of the SL. From a linguistic point of view what is missing is any analysis of the internal structure of the SL, particularly the grammatical relationships between the principal parts of the sentences.

Before leaving the direct translation method, it should be noted that it continues to some extent in many unidirectional bilingual systems. In certain circumstances this approach is still valid today -traces of the direct approach are found in systems such as Meteo - a successful MT system to translate weather forecasts.

The most well-known direct MT systems for Indian languages are Anusaaraka developed by IIT Kanpur and now being continued at IIT Hyderabad, and Hindi to Punjabi MT system developed at the Punjab University, which achieved an accuracy of 95%.

Indirect architectures

The failure of the first generation systems led to the development of more sophisticated linguistic models for translation. In particular, there was increasing support for the analysis of SL texts into some kind of intermediate representation - a representation of its “meaning” in some respect -which could form the basis of generation of the TL text. This is in essence the indirect method, which has two principal variants.

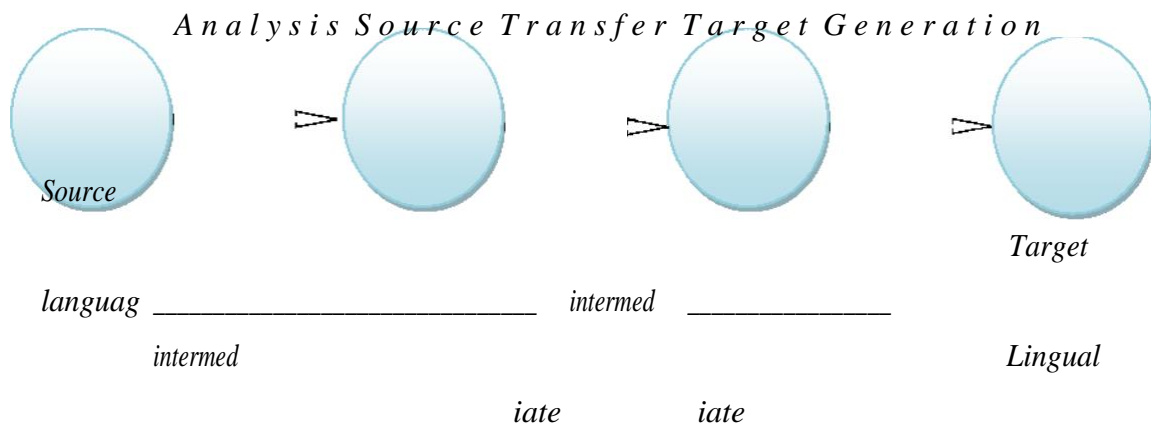
The intermediate representation includes all information necessary for the generation of the TL text without “looking back” to the original text. The representation is thus a projection from the SL text and at the same time acts as the basis for the generation of the TL text; it is an abstract representation of the TL text as well as a representation of the SL text. The method is interlingual in the sense that the representation is neutral between two or more languages. In the past, the intention or hope was to develop an interlingual representation which was truly “universal” and could thus be intermediary between any natural languages.

The advantage is that the addition of a new language to the system entails the creation of just two new modules: an analysis grammar and a generation grammar. By adding one analysts module, e.g. a German analysis grammar, the number of translation directions is increased from two (English to French, and French to English) to four (by the addition of German to French and German to English). The inclusion of another generation module, a German generation grammar, brings a further two pairs (English to German and French to German). The addition of two further modules for, say, Spanish, would increase the number of language pairs by another six (English, French and German into Spanish, and Spanish into English, French and German), and so on exponentially.

While the addition of new languages may appear easy in an interlingual system, there are major disadvantages: the difficulties of defining an interlingua, even for closely related languages (e.g. the Romance languages: French, Italian, Spanish, Portuguese). A truly “universal” and language-independent interlingua has defied (challenged) the best efforts of linguists and philosophers from the seventeenth century onwards. At present, interlingual systems are less ambitious.

There are several MT systems in India developed on interlingua approach. Anglabharti developed at IIT Kanpur uses an intermediate structure Pseudo Lingua for Indian Languages. AnglaHindi is an extension of Anglabharti which uses rule-based, example-based and statistics-based methods¹ to obtain translation for frequently encountered noun and verb phrases. Another notable system is the Universal Natural Language English to Hindi Machine Translation system developed at IIT Bombay, which uses a Universal Natural Language (UNL) as propose by United Nations University. Currently, work is in progress for English to Marathi and English to Bengali UNL systems.

The second variant of the indirect architectures is called the *transfer* method. In fact, all translation systems involve “transfer” of some kind, the conversion of SL text or representation into TL text or representation. Dictionary directly converts source into target whenever a sentence matches one of the transfer rules. SL dictionary, TL dictionary and a bilingual dictionary is used for this purpose. The translation mechanism is carried out in three phases. Firstly, the SL is converted into intermediate representation (Analysis) which is subsequently converted into TL representation in the second phase (Transfer). The third phase involves Generation of the final target language.



Unlike those in interlingual systems the transfer representations are language-dependent: the result of analysis is an abstract representation of the SL text, the input to generation is an abstract representation of the TL text. The function of the bilingual transfer modules is to convert SL (intermediate) representations into TL (intermediate) representations. Since

these representations link separate modules (analysts, transfer, generation), they are also frequently referred to as interface representations.

In comparison with the interlingua type of multilingual system there are clear disadvantages in the transfer approach. The addition of a new language involves not only the two modules for analysis and generation, but also the addition of new transfer modules, the number of which may vary according to the number of languages in the existing system: in the case of a two-language system, a third language would require four new transfer modules.

Indian MT systems based on transfer architecture include Mantra, developed by Applied Artificial Intelligence group of CDAC Pune. Another translation system is the Matra system also developed by CDAC Pune. There is also Shakti MT system developed by IIIT Hyderabad and IISC Bangalore which works by combining rule-based and statistical methods. Anubaad is another MT system which is a hybrid system using *ngram* method for POS tagging and works at sentence level developed at CDAC Kolkata. Another notable MT system is the English to Kannada, developed at the University of Hyderabad which uses Universal Clause Structure Grammar Formalism. Sampark, another translation system developed at IIIT Hyderabad, translates Indian Languages to Indian Languages which is based on Paninian framework. The Sampark is deployed in cloud environment which reduces the deployment time and distributed system environment which increases the throughput.

2.2. Paradigms

The architectural basis of the system is only one of many axes along which one might compare MT systems. Another important axis of comparison is that of research paradigm. It is important to understand the difference between the type of architecture and the type of paradigm: one does not presuppose the other. Architecture refers to the

actual processing design (i.e., direct, transfer, and interlingua), whereas the paradigm refers to informational components that aid the processing design (knowledge-based, example-based, statistics-based, etc.)

Research paradigms are classified into three categories: 1) those that propose to rely most heavily on linguistics techniques (linguistics paradigms); 2) those that do not use any linguistics techniques (non-linguistics); 3) those that use a combination of two (hybrid).

Linguistic paradigms

Until recently, most MT researchers studied linguistic-based MT, i.e., translation on the basis of principles that are well-grounded in linguistic theory. Systems based on linguistic theory strive to use the rules of syntax, lexicon, and semantics to produce an appropriate TL realization of the SL sentence.

Constraint-based MT

CBMT techniques have shown up several different MT approaches. For ex., Shake-and-Bake approach demonstrates the full utility of constraint application. Initial approach in CBMT was using constraints on combination of lexical items, i.e., the Lexical Functional Grammar MT system. This system translates bidirectionally based on lexical functional grammar (LFG). In the LFG formalism, f-structure (functional structure) is a fundamental component of the translation.

Because the LFG MT system is based on construction-specific representations, the mapping operations required in the transfer must be performed by transfer equations that relate source-and target-language f-structures. The LFG MT framework makes an association between the syntactic structure and the f-structure using a set of mediating selectional constraints that are encoded as lexical entries.

The disadvantage of this approach is that the f-structure is tightly coupled with the syntactic structure of the language; thus if a particular concept can be syntactically expressed in more than one way, there will be more than one f-structure in this framework. A more refined version of this approach is currently used in the Verbmobil MT project.

Knowledge-based MT

KBMT is a paradigm adopted by developers of numerous US systems including Pangloss at Carnegie Mellon's Center for MT, ULTRA at New Mexico State's Computing Research Lab, Cyc-NL at CYCORP, Japangloss and GAZELLE (Univ of S. California), UNICON (Univ. of Pennsylvania), etc.

KBMT paradigm is concentrated on the development of knowledge intensive morphological, syntactic and semantic information for lexicon. Tools have been developed for creating ontologies and have provided a framework for automatic lexical acquisition. In general, the focus of research in KBMT has been on the development of underlying knowledge representation concepts. Little attention has been devoted to providing a systematic "linking" of these concepts to the syntax; thus, KBMT is complementary to the (less knowledge intensive) Lexical-based MT and Principle-based MT paradigms described later.

In its own, the KBMT provides higher-quality, fully-automated translations; however, the domains used for this approach are narrow and well-defined since the required amount of information would otherwise be large and cost-prohibitive with respect to indexing and searching. Disadvantage in this paradigm is that there no single systematic mapping between a concept and its surface realization.

Lexical-based MT

LBMT overlaps heavily with several other approaches including RBMT, PBMT and SBMT. In general, a LBMT system refers to any system that supplies rules for relating the lexical entries of one language to the lexical entries of another language. Several researchers have adopted the lexical-based paradigm, but at different degrees of generality.

One such system is LTAG (English-French and vice versa). System uses a transfer approach that uses synchronous tree-adjoining grammars to map shallow TAG derivations from one language into another. The mapping is performed by means of a bilingual lexicon which directly associates source and target trees through links between lexical items and their arguments. Roughly each bilingual entry contains a mapping between a SL sentence and a TL sentence.

One disadvantage of this approach is that it requires entire trees to be stored in the transfer dictionary for each source-to-target pair. This is significantly burdensome as the number of SL and TL begin to add up.

Rule-based MT

The RBMT paradigm is associated with systems that rely on different linguistic levels of rules for translation between the SL and TL. The prototypical example of such a system is Roseta which divides translation rules into two categories: 1) S-rules which are “non-meaningful rules” that map lexical items to syntactic trees; and 2) M-rules which are “meaning preserving rules” that map between syntactic trees to underlying meaning structures. In cases of syntactic mismatches between SL and TL representations, a “switch-rule” is invoked and normal processing is interrupted, control is then passed to a module that derives a new category that takes over the role of syntactic head.

One problem with this approach is that it leaves open the question of how grammar driven interrupts (that occur in case of mismatches) interact with idiosyncratic requirements of individual lexical items.

Principle-based MT

PBMT paradigm has been developed by a number of researchers. PBMT is an alternative to RBMT in which rules are replaced by a small set of principles that cover morphological, grammatical, and lexical phenomena in a general fashion. An example of a PBMT paradigm is the PRINCITRAN MT system which is based on syntactic principles of Government-Binding Theory and lexical-semantic principles of Lexical-Conceptual Structure. This system was originally designed for Korean-English MT, lexicons for Spanish and Arabic were subsequently developed for related NLP applications (ex., foreign language tutoring). In this system, structure building is deferred until underlying description satisfies a number of linguistic principles that are parametrized for ease of porting to new languages. The processing mechanism is language independent and accommodates structurally different languages (ex., head-initial vs head-final) with equally efficient run times.

PBMT is considered to be a complementary to the KBMT and EBMT (example-based) paradigms in that it provides broad coverage of many different linguistic phenomena, but lacks deeper knowledge about the translation domain. There were efforts to combine the benefits of PBMT and KBMT techniques in a collaborative large-scale Chinese-English MT effort between the University of Maryland and New Mexico State University.

Brief History of Machine Translation Development

The idea of creating mechanical dictionaries to solve problem of language barriers was first suggested by Descartes and Leibniz in the 17th century. The aim was to create a universal unambiguous language, which would be based on logical principles and iconic symbols. Later, there were many other proposals for international languages (for example, Esperanto), but few attempted to mechanize translation until the middle of the last century. Scholarly research defines several stages of MT development.

Lexical transfer

Identifies three ways in which negative **lexical transfer** from Spanish to English may occur: overextension of analogy (false cognates), substitution errors, and interlingual/intra-lingual interference.

The Role of Computational Linguistics (CL) in Language Teaching

Computational linguistics (CL) is the application of computer science to the analysis, synthesis and comprehension of written and spoken language. Computational linguistics is used in instant machine translation, speech recognition (SR) systems, text-to-speech (TTS) synthesizers, interactive voice response (IVR) systems, search engines, text editors and language instruction materials. The interdisciplinary field of study requires expertise in machine learning (ML), deep learning (DL), artificial intelligence (AI), cognitive computing and neuroscience.

A computational understanding of language provides human beings with insight into thinking and intelligence. Computers that are linguistically competent not only help facilitate human interaction with machines and software, but also make the textual and other resources of the internet readily available in multiple languages. Business goals of computational linguistics include:

- Translating text from one language to another.
- Retrieving text that relates to a specific topic.
- Analyzing text or spoken language for context, sentiment or other affective qualities.
- Answering questions, including those that require inference and descriptive or discursive answers.
- Summarizing text.
- Building dialogue agents capable of completing complex tasks such as making a purchase, planning a trip or scheduling maintenance.

- Creating chatbots capable of passing the Turing Test.

Most work in computational linguistics – which has both theoretical and applied elements – is aimed at improving the relationship between computers and basic language. It involves building artifacts that can be used to process and produce language. Building such artifacts requires data scientists to analyze massive amounts of written and spoken language in both structured and unstructured formats.

Typically, computational linguists are employed in universities, governmental research labs or large enterprises. In the private sector, vertical companies, like Caterpillar typically employ computational linguists to authenticate the accurate translation of technical manuals. Tech software companies, such as Microsoft, typically hire computational linguists to work on natural language processing (NLP), helping programmers to create voice user interfaces (VUIs) that will eventually allow humans to communicate with computing devices as if they were another person.

More job opportunities exist for linguistics experts to help developers improve Internet search engines, build virtual assistants and integrate speech recognition with other language processing techniques. Demand is also growing for computational linguists in the public sector as government grapple with the continuous growth of unstructured data.

Although the concept of CL is often associated with AI, CL pre-dates AI's development, according to the Association for Computational Linguistics.

Building Search Engines

The web Each step in the ascent of corpora was underwritten by computers becoming orders of magnitude larger (in disk and memory capacity), faster and cheaper. In the last decade, the greatest development in computing has been the web. It contains, at a conservative estimate, 100 billion words of English (1000 times more than the BNC)

as well as lesser, but still very large, quantities of many other languages (Grefenstette and Nioche 2000), and conveniently supplies a mechanism for delivering the data it holds.

It is large enough for extensive exploration of word combinations, for corpus linguistics for many ‘smaller’ languages, and to contain very, very many text types and sublanguages. Much of the data is classified, explicitly (with keywords or in Yahoo or similar topic hierarchies) or implicitly (eg through the pages a page links to, which may be classified, or simply through the vocabulary used) and this can support further uses of the data (e.g. Agirre and Martinez 2000). Existing search engines More and more people are starting to use the web for language research, for everything from spell-checking to finding academic papers. But the obvious way to use it – through existing search engines – is limiting and frustrating for linguistic research (see above) and can easily lead to wasted effort, distorted results and bad science. Because it is so large, downloading large parts of it is non-trivial. Resnik and Smith (2003) address the issue by using the Internet Archive, a snapshot of the web made available for research, and this is an interesting avenue to pursue, but involves a further organization with its own concerns, priorities and technical and legal constraints. Our proposal may be contrasted with the approach taken by Webcorp (<http://www.webcorp.org.uk>). While their goal is similar to ours, they proceed though a meta-search engine which takes user input, converts it into a query for Google and other engines, and then packages the data returned by the engines as a concordance. They remain dependent on other search engines and whatever distortions they introduce. Anecdotal evidence is that the distortions are considerable, with anecdotal frequencies given for “pages containing the words x and y” varying wildly; the search engines’ priority is to respond fast, speed mattering far more to most search engine users than the accuracy of frequency counts. Existing search engines cannot be depended on for reliable lexical statistics. Much web search engine technology has been developed with reference to HLT.

The prototype for Altavista was developed in a joint project between Oxford University Press, whose goal was to explore methods for corpus lexicography, and DEC, who used the lexicographic corpus as a very large database requiring fast access for a

range of queries. Language identification algorithms, now widely used in web search engines, were developed as HLT technology. This project offers the prospect of a 'homecoming' for web search, with it now feeding the hand that fostered it. Components

The components of the LSE are 1. web crawler 2. filter/classifier (o) 3. linguistic processor (o) 4. database 5. statistical summariser (o) 6. user interface. The three items marked 'o' are optional, in that a search engine could meet linguists' needs, to the extent of giving linguistics-oriented web access, without them. However they are central to linguists getting more out of the web, and are key to this proposal.

1. Web Crawler

Setting up a web crawler to crawl the whole web will involve establishing a very high bandwidth web connection, and negotiating for it to bypass all caches. For regular search engines, it is critical to be up-to-date, so they crawl the web very frequently and this places high performance demands on their hardware.

Information retrieval system NLP

Natural Language processing in textual **information retrieval**. Indexing the collection of documents: in this phase, **NLP** techniques are applied to generate an index containing document descriptions. Normally each document is described through a set of terms that, in theory, best represents its content.

"Natural Language Processing" (NLP) as a discipline has been developing for many years. It was formed in 1960 as a sub-field of Artificial Intelligence and Linguistics, with the aim of studying problems in the automatic generation and understanding of natural language.

At first its methods were widely accepted and successful. However, when applied in controlled environments and with a generic vocabulary, many problems arose. Among those problems were polysemy and synonymy.

In recent years contributions to this field have improved substantially, allowing for the processing of huge amounts of textual information with an acceptable level of efficacy. An example of this is the application of these techniques as an essential component in web search engines, in automated translation tools or in summary generators [Baeza-Yates, 2004].

This article aims to review the main characteristics of natural language processing techniques, focusing on its application in information retrieval and related topics. Specifically, in the second section we will study the different problems in automatic natural language processing; in the third section we will describe the key methodologies of NLP applied in information retrieval; and in the fourth section we will state several fields of research related to information retrieval and natural language processing; finally we present the conclusions and an annexe (Annexe 1) showing some of the particular aspects of NLP in Spanish.

Natural language, understood as a tool that people use to express themselves, has specific properties that reduce the efficacy of textual information retrieval systems. These properties are linguistic variation and ambiguity. By linguistic variation we mean the possibility of using different words or expressions to communicate the same idea. Linguistic ambiguity is when a word or phrase allows for more than one interpretation. Both phenomena affect the information retrieval process, even though in different ways. Linguistic variation provokes document silence, that is, the omission of relevant documents that fulfil information needs, because the same terms were not used as those found in the document. Ambiguity, on the other hand, implies document noise, or the inclusion of non-meaningful documents, since documents were retrieved that used the same term but with a different meaning. These characteristics make automated language processing considerably difficult. The following is a set of examples that show the repercussions of these phenomena in information retrieval:

At a morphological level, the same word may play different morph-syntactic roles relative to the context in which they appear, causing ambiguity problems (example 1).

Example 1. A notebook was the present that his wife gave him when all of us were present at the party.

In this case, the word "present" acts both as an adjective and noun, and with different meanings.

At a syntactic level, focusing on the study of established relations between words to form larger linguistic units, phrases and sentences, ambiguities are produced as a consequence of the possibility of associating a sentence with more than one syntactic

structure. On the other hand, this variation supposes the possibility of expressing the same idea, but changing the order of the sentence's syntactic structure. (example 2).

Example 2. He ate the chocolates on the plane.

This example could mean that "He ate the chocolates that were in the plane" or that "He ate the chocolates when he was flying in the plane."

At a semantic level we study the meaning of a word and sentence by studying the meaning of each of the words in it. Ambiguity is produced because a word can have one or various meanings, which is known as polysemy (example 3).

Example 3. Paul was reading a newspaper in the bank.

The term "bank" could refer to a financial institution or a mound.

And we must also keep in mind lexical variation which refers to the possibility of using different terms for the same meaning, that is, a synonymy (example 4):

Example 4: Car / Auto / Automobile.

At a pragmatic level, based on a language's relationship to its context, we often can not use a literal and automated interpretation of the terms used. In specific circumstances, the sense of the words in the sentence must be interpreted at a level that includes the context in which the sentence is found. (example 5).

Example 5. Give me a break.

Here we are asking for rest from work, or we could be asking the perceiver to leave us alone.

Another important topic is ambiguity provoked by an anaphora, for example, the presence of pronouns and adverbs that refer to something that was previously mentioned (example 6).

Example 6. It was terrible for him she had not to manipulate it.

Who is he? And she? What was not manipulated? It is impossible to understand this sentence out of context.

All of these examples demonstrate the complexity of language, and that any automated processing is not easy or obvious.

Natural Language Processing in Textual information Retrieval

As the reader has probably already deduced, the complexity associated with natural language is especially key when retrieving textual information [Baeza-Yates, 1999] to satisfy a user's information needs. This is why in Textual Information Retrieval, NLP techniques are often used [Allan, 2000] both for facilitating descriptions of document content and for presenting the user's query, all with the aim of comparing both descriptions and presenting the user the documents that best satisfy their information needs.

In other words, a textual information retrieval system carries out the following tasks in response to a user's query (image 1):

1. Indexing the collection of documents: in this phase, NLP techniques are applied to generate an index containing document descriptions. Normally each document is described through a set of terms that, in theory, best represents its content.
2. When a user formulates a query, the system analyses it, and if necessary, transforms it with the hope of representing the user's information needs in the same way as the document content is represented.
3. The system compares the description of each document with that of the query, and presents the user with those documents whose descriptions are closest to the query description.
4. The results are usually listed in order of relevancy, that is, by the level of similarity between the document and query descriptions.

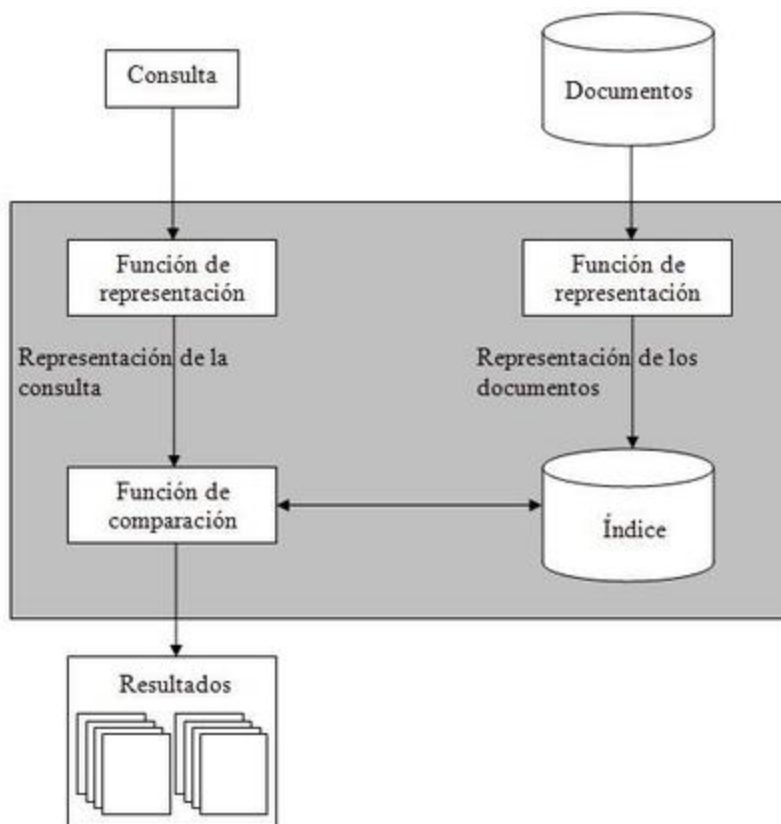


Image 1: The architecture of an information retrieval system

As of now there are no NLP techniques that allow us to extract a document's or query's meaning without any mistakes. In fact, the scientific community is divided on the procedure to follow in reaching this goal. In the following section we will explain the functions and peculiarities of the two key approaches to natural language processing: a statistical approach and a linguistic focus. Both proposals differ considerably, even though in practice natural language processing systems use a mixed approach, combining techniques from both focuses.

Statistical processing of natural language

Statistical processing of natural language [Manning, 1999] represents the classical model of information retrieval systems, and is characterised from each document's set of key words, known as the terms index.

This is a very simple focus based on the "bag of words." In this approach, all words in a document are treated as its index terms. Moreover, each term is assigned a weight in

function of its importance, usually determined by its appearance frequency within the document. This way the word's order, structure, meaning, etc, are not taken into consideration.

These models are then limited to pairing the documents' words with that of the query's. Its simplicity and efficacy has become the most commonly used contemporary models in textual information retrieval systems.

This document processing model involves the following stages:

a) Document pre-processing: fundamentally consisting in preparing the documents for its parameterisation, eliminating any elements considered as superfluous.

b) Parameterisation: a stage of minimal complexity once the relevant terms have been identified. This consists in quantifying the document's characteristics (that is, the terms).

Below we will illustrate their function using this paper's first paragraph as an example, assuming that it is XML tagged. So the document on which we would apply the pre-processed and parameterisation techniques would be the following:

```
<document document_ID="000127" source=http://www.hipertext.net>
  <title>
Processing Natural Language in Textual Information Retrieval and related topics.
  </title>
  <body>
1. Introduction
"Natural Language Processing" (NLP) as a field has been developing for many
years. It was formed in 1960 as a sub-field of Artificial Intelligence and
Linguistics, with the aim of studying problems in the automated generation and
understanding of natural language.
...
  </body>
</document>
```

Document pre-processing consists of three basic phases:

1. Elimination of the elements in the document that are not for indexing (stripping), such as some document tags or headers (example 5).

```
Processing Natural Language in Textual Information Retrieval and related topics.
1. Introduction
"Natural Language Processing" (NLP) as a field has been developing for many
years. It was formed in 1960 as a sub-field of Artificial Intelligence and
Linguistics, with the aim of studying problems in the automated generation and
understanding of natural language.
...
```

Example 5. Document without headers or tags

2. Text standardising, consisting in homogenising the whole text in the complete collection of documents to be worked on, including the consideration of capitalised or non-capitalised terms, checking specific parameters like numerals or dates; abbreviations or acronyms, eliminating empty words by applying the lists of functional words (prepositions, articles, etc.) identifying N-Grams, (the example's terms and underlined terms). (Example 6).

```
processing natural language in textual information retrieval and related topics
StringNumber introduction
Processing natural language NLP has been developing for StringNumber sub-area
linguistic artificial intelligence aim study problems in the automated generation
and understanding of natural language.
...
```

Example 6. Standardised document

3. Stemming terms is a linguistic process that attempts to determine the base (lemma) of each word in a text. Its aim is to reduce a word to its root, so that the key words in a query or document are represented by their roots instead of the original words. The lemma of word is its basic form along with its inflected forms. For example, "inform" could be the lemma of "information" or "inform." The stemming process (example 7) is carried out by using algorithms that can represent the different variants of a term at once, while also reducing the amount of vocabulary and as a consequence improving the capacity of storage in systems, as well as document processing time. However, these algorithms have the -inconvenience of sometimes not grouping words that should be grouped, and vice versa: erroneously presenting words as equals.

```
natural language text information retrieval similar topic
StringNumber introdu
Process natural language NLP has been developing for StringNumber sub-area
linguist artificial intelligence linguistic aim study problem in generate automatic
natural language understand
...
```

Example 7. Documents with stemmed terms

Parametrising documents consists in assigning a weight to each one of the relevant terms associated to a document. A term's weight is usually calculated as a function of its appearance frequency in the document, indicating the importance of these terms as the document's content description (example 8).

Related	1	linguist	1
area	1	from	1
automate	1	aim	1
understand	1	NLP	1
develop	1	problem	1
in	1	process	2
field	1	information retrieval	1
study	1	StringNumber	--
generate	1	subarea	1
artificial intelligence	1	text	1
introd	1	years	1
many	1		
natural language	3	[...]	

Example 8. Fragment of a parametrised document (see how the frequencies of each term changes as the quantification of the remaining terms in the document continues)

One of the most often used methods to estimate the importance of a term is the TF.IDF system (Term Frequency, Inverse Document Frequency). It is designed to calculate the importance of a term relative to its appearance frequency in a document, but as a function of the total appearance frequency for all of the corpus' documents. That is, the fact that a term appears often in one document is indicative that that term is representative of the content, but only when that term does not appear frequently in all documents. If it appeared frequently in all documents, it would not have any discriminatory value (for example, it would be absurd to represent the content of a document in a recipe database by the frequency of the word food, even though it appears often).

Finally, and as we have already mentioned, we must describe two commonly used techniques in the statistical processing of natural language:

a) Detecting N-Grams: this consists in identifying words that are usually together (compound words, proper nouns, etc.) to be able to process them as a single conceptual unit. This is usually done by estimating the probability of two words that are often together make up a single term (compound). These techniques attempt to identify compound terms such as "accommodation service" or "European Union."

b) Stopwords lists: a list of empty words in a terms list (prepositions, determiners, pronouns, etc.) considered to have little semantic value, and are eliminated when found in document, leaving them out of the terms index to be analysed. Deleting all of these terms avoids document noise problems and saves on resources, since in documents few elements are repeated frequently.

Linguistic processing of natural language

This approach is based on the application of different techniques and rules that explicitly encode linguistic knowledge [Sanderson, 2000]. The documents are analysed through different linguistic levels (as previously mentioned) by linguistic tools that incorporate each level's own annotations to the text. Below we show the different steps to take in a linguistic analysis of documents, even though not all systems use them.

The morphological analysis is performed by taggers that assign each word to a grammatical category according to the morphological characteristics found.

After having identified and analysed the words in a text, the next step is to see how they are related and used together in making larger grammatical units, phrases and sentences. Therefore a syntax analysis of the text is performed. This is when parsers are applied: descriptive formalism that demonstrate the text's syntax structure. The techniques used to apply and create parsers vary and depend on the aim of the syntax analysis. For information retrieval it is often used for a superficial analysis aiming to only identify the most meaningful structures: nominal sentences, verbal and prepositional sentence, values, etc. This level of analysis is usually used to optimise resources and not slow down the system's response.

From the text's syntax structure, the next aim is to obtain the meaning of the sentences within it. The aim is to obtain the sentence's semantic representation from the elements that make it up.

One of the most often used tools in semantic processing is the lexicographic database WordNet. This is an annotated semantic lexicon in different languages made up of synonym groups called synsets which provide short definitions along with the different semantic relationships between synonym groups.

WordNet Search - 3.0 - [WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for:

Display Options:

Key: "S." = Show Synset (semantic) relations, "W." = Show Word (lexical) relations

Noun

- [S:](#) (n) [car](#), [auto](#), [automobile](#), [machine](#), [motorcar](#) (a motor vehicle with four wheels; usu *get to work*"
- [S:](#) (n) [car](#), [railcar](#), [railway car](#), [railroad car](#) (a wheeled vehicle adapted to the rails of re
- [S:](#) (n) [car](#), [gondola](#) (the compartment that is suspended from an airship and that carries
- [S:](#) (n) [car](#), [elevator car](#) (where passengers ride up and down) "*the car was on the top*
- [S:](#) (n) [cable car](#), [car](#) (a conveyance for passengers or freight on a cable railway) "*they*

[WordNet home page](#)

Figure 2: An example of semantic information provided by WordNet. <http://wordnet.princeton.edu/perl/webwn>

Corpus Linguistics

Corpus (corpora) is a large collection of written or spoken texts available in machine readable form accumulated in scientific way to represent a particular variety or use of a language. We hope to have shown that corpus resources and methods have a great potential to improve pedagogical practice and that corpora can be used in a number of ways, indirectly to inform teaching materials and reference works, or directly as language learning tools and repositories for the design of data-intensive teaching activities.

The growth of corpus research in recent years is evidenced not only by the growing number of new corpora but also by their wider variety. Various “specialized corpora” have recently been created. One of them is the “learner corpus”, which is a collection of the language spoken or written by non-native speakers. The primary purpose of learner corpora is to offer researchers and language teaching professionals resources for their research. In order to develop a curriculum or pedagogy of language

teaching, it would be beneficial to have interlanguage data so that researchers can scientifically describe the characteristics of each developmental stage of their interlanguage.

Although there are a lot of advantages of error-annotated learner corpora, found some difficulties in designing an error tagset that covers important features of learner errors. The current version of our error tagset targets morphological, grammatical, and lexical errors, and found it can help to successfully assess to what extent learners can command the basic language system, especially grammar. However, also found that it is not sufficient to measure learners' communicative skills. In order to determine how the current error tagset should be extended to cover more communicative aspects of learner language.

Type of corpus

A corpus is always designed for a particular purpose, and the type of corpus will depend on its purpose. There are some commonly used corpus types :

Specialized Corpus

A corpus of a text of particular type, such as a newspaper, Editorials, geography text book, academic articles in a particular subject, lectures, casual, conversation, essays written by students etc .it aims to be representative of a given type of text. It is used to investigate a particular type of language. Researchers he often collect their own specialized corpora to reflect kind of language they want to investigate. There is no limit to the degree specialization involved, but The parameters are set to limit the kind of text Included . For example, a corpus might be restricted to frame, consisting of text. Form a particular century ,or to a social setting, such as conversations taking place in a bookshop, or to a given topic, such as newspaper articles dealing with the European union. Some well-known specialized corpora include the 5 million word Cambridge and Nottingham corpus of Discourse in English (CANCODE) (informal registers of British

English) and the Michigan corpus of Academic spoken English (MICASE) (spoken registers in a US academic setting)

General corpus

A corpus of texts of many types. It may include written or spoken language, or both and may include texts produced in one country. It is unlikely to be representative of any particular 'whole', but will include as wide a spread of texts as possible. A general corpus is usually much larger than a specialized corpus. It may be used to produce reference materials for language learning or translation, and it is often used as a baseline in comparison with more specialized corpora. Because of this second function it is also sometimes called a reference corpus. Well known general corpora include the British National corpus (100 million words) and the Bank of English (400 million words in January 2001), both of which comprise a range of sub corpora from different sources. Much earlier general corpora were the LOP corpus, consisting of written British English, and the Barrow corpus, consisting of written American English, both compiled in the 1960s and comprising 1 million words each.

Comparable Corpora

Comparable Corpora (two more) in different languages (e.g. English and Spanish) different varieties of a language (e.g. Indian and Canadian English) they are designed along the same line, for example they will contain the same proportions of newspaper text, novels, casual conversation, and so on. Comparable corpora of varieties of the same language can be used by translators and by learners to identify differences and equivalences in each language. The ICE corpora (international corpus of English) are comparable corpora of 1 million words each of different varieties of language.

Parallel corpus

Parallel corpus (two or more) corpora in different languages, each containing text that have been translated from one language into the other (e.g. novels in English that has been translated into Spanish, and one in Spanish that has been translated into English) or text that have been produced simultaneously in two or more languages e.g. European Union regulations, which are published in all the official languages of the EU. They can be used by translators and by learners to find potential equivalent expressions in each language and to investigate differences between languages.

The Concept of Corpus Linguistics in Language Teaching and Learning

This is relationship between corpus linguistics (CL), language teaching (LT) and provides an overview of the most important pedagogical applications of corpora. Corpus (corpora) is a large collection of written or spoken texts available in machine readable form accumulated in scientific way to represent a particular variety or use of a language. Corpus evidence has not only been used in linguistic research but also in the teaching and learning of languages probably a use that “the compilers of corpora may not have foreseen” (Johansson 2007).

There is now a wide range of fully corpus-based reference works (such as dictionaries, glossaries and grammars) available to learners and teachers, and a number of dedicated researchers and teachers have made concrete suggestions on how concordances and corpus-derived exercises could be used in the language teaching classroom, thus significantly “enriching the learning environment” (Aston 1997, 51). We hope to have shown that corpus resources and methods have a great potential to improve pedagogical practice and that corpora can be used in a number of ways, indirectly to inform teaching materials and reference works, or directly as language learning tools and repositories for the design of data-intensive teaching activities. The aim of this paper concerned the importance of the corpus linguistics in language teaching and learning

Corpus

Corpus (corpora) is a large collection of written or spoken texts available in machine readable form accumulated in scientific way to represent a particular variety or use of a language. It serves as an authentic data for linguistic and other related studies. The size, text type, organization, accessing method, etc. are some of the basic features of a corpus which have to be carefully decided while generating a corpus. There are different types, which are again determined by the purpose for which the corpus is built.

We hope to have shown that corpus resources and methods have a great potential to improve pedagogical practice and that corpora can be used in a number of ways, indirectly to inform teaching materials and reference works, or directly as language learning tools and repositories for the design of data-intensive teaching activities. The main focus of corpus linguistics is to discover patterns of authentic language use through analysis of actual usage. The aim of a corpus based analysis is not to generate theories of what is possible in the language, such as Chomsky's phrase structure grammar which can generate an infinite number of sentences but which does not account for the probable choices that speaker actually make. Corpus linguistics' only concern is the usage patterns of the empirical data and what that reveals to us about language behavior.

The Error - Tagged Corpus

After having explained some of the main features to have in mind when developing a work on corpus linguistics, attention is focused on corpus mark-up. This characteristic of the corpus can be defined as the set of codes added to the text so as to obtain information about it and be able to manipulate the processing of the document. Corpus mark-up could be a discouraging aspect when one has a first contact with corpus linguistics; in fact, watching the amounts of 'incomprehensible' symbols, numbers and letters around a text provokes a feeling of ignorance but once we are able to understand the codes, we can appreciate the great value of this system.

A term that is related to corpus mark-up is corpus annotation. If the mark-up of a corpus provides great advantages in terms of linguistic research studies, the advantages increase when a corpus is annotated as the information added to the text is linguistic, e.g. it provides information of the grammar class of words. Tagging can be automatic (the computer program tags the corpus) or manual (there is human intervention in the tagging with the aid of a computer program). Hunston (2002: 18) points out the level of accuracy achieved depending on the automatic or manual tagging, and states that even though in

the first case a corpus can be totally tagged, the level of accuracy is lower than that obtained through manual tagging, however this is only feasible when the corpus is small.

Error-tagging is a specific type of annotation and the corpora on which error-tagging is done is normally associated to second language learners. May be the best exploitation of an error-annotated corpus is to observe the frequency of errors made by second language learners and their type. More interesting information is obtained when the study of errors is done with an emphasis on a specific language background, like in this case, where the purpose is to find about the frequency of errors and error types made by learners of Tamil with non-native.

In general, tagging can be done in two ways: rule-based or probabilistic; rule-based respects the rules of grammar written into the tagger, probabilistic tagging is “based on the statistical likelihood that a given tag will occur in a given context” (Meyer, 2002: 88). However, for the case of error-tagging the task is more than complicated since determining error patterns can become very arbitrary, and on the other hand, the frequencies of error differ from one group of learners (with a specific language background) to another. That is why the effort of error-tagging learner corpora from ICLE is greatly appreciated; otherwise this analysis would not have been possible.

Corpus linguistics and language teaching

There is now a wide range of fully corpus-based reference works (such as dictionaries, glossaries, school text books and grammars) available to learners and teachers, and a number of dedicated researchers and teachers have made concrete suggestions on how concordances and corpus-derived exercises could be used in the language teaching classroom, thus significantly enriching the learning environment” (Aston 1997, 51). Indicative of the popularity of pedagogical corpora use and the need for research in this area is the considerable number of books and edited collections of some of which are the result of the successful “Teaching and Language Corpora” (TaLC) conference series that have recently been published on the topic of this article or which bear a close

relationship to it (cf. Ädel 2006; Aston 2001; Aston/Bernardini/Stewart 2004; Bernardini 2000a; Botley et al. 1996; Braun/Kohn/Mukherjee 2006; Burnard/McEnery 2000; Connor/Upton 2004; Gavioli 2006; Ghadessy/Henry/Roseberry 2001; Granger/Hung/Petch-Tyson 2002; Hidalgo/Quereda/Santana 2007; Hunston 2002; Kettemann/Marko 2002; Mukherjee 2002; Nesselhauf 2005; Partington 1998; Römer 2005a; Schlüter 2002; Scott/Tribble 2006; Sinclair 2004a; Wichmann et al. 1997).

This relationship is a dynamic one in which the two fields greatly influence each other. While LT profits from the resources, methods, and insights provided by CL, it also provides important impulses that are taken up in corpus linguistic research. The requirements of LT hence have an impact on research projects in CL and on the development of suitable resources and tools. It will also discuss further possible effects of CL on LT and of LT on CL, and highlight some future tasks for researchers and practitioners in the field.

Corpora and reference works and teaching materials

The results of the above mentioned corpus-coursebook comparisons do not only inform the language teaching curriculum but also help with decisions about the presentation of items and structures in reference works and teaching materials. Research on general corpora has exerted a huge influence on reference publishing and has led to a new generation of dictionaries and grammar books. Nowadays, “people who have never heard of a corpus are using the products of corpus research.” (McEnery/Xiao/Tono 2006, 97) In the context of ELT (English Language Teaching), the publications in the Collins CO-BUILD series constitute a major achievement. Based on real English and compiled with the needs of the language learner in mind, the COBUILD dictionaries, grammars, usage guides, and concordance samplers (cf. Capel 1993; Carpenter 1993; Goodale 1995; Sinclair et al. 1990; Sinclair et al. 1992; Sinclair et al. 2001) offer teachers and learners more reliable information about the English language than

any of the more traditional reference grammars or older non-corpus-based dictionaries.

Two major advantages of the COBUILD and other corpus-based reference works for learners, e. g. those published in the past few years by Longman, Macmillan, OUP and CUP (cf. e. g. Abbs/Freebairn 2005; Biber/Leech/Conrad 2002; Hornby 2005; Peters 2004; Rundell et al. 2002) are that they incorporate corpus-derived findings on frequency distribution and register variation, and that they contain genuine instead of invented examples. Particularly worth mentioning here is the student version of the entirely corpus-based *Longman Grammar of Spoken and Written English* (Biber et al. 2002). The importance of presenting learners with authentic language examples has been stressed in a number of publications (cf. deBeaugrande 2001; Firth 1957; Fox 1987; Kennedy 1992; Römer 2004b, 2005a; Sinclair 1991, 1997). Kennedy (1992, 366), for instance, cautions that “invented examples can present a distorted version of typicality or an over-tidy picture of the system”, and Sinclair (1991, 5) calls it an “absurd notion that invented examples can actually represent the language better than real ones”. Thanks to the ‘corpus revolution’, the language learner can today choose from a range of reference works that are thoroughly corpus-based and that offer improved representations of the language she or he wants to study.

Another branch of general corpora research that has exerted some influence on the design of reference works and, to a lesser extent, teaching materials is the area of phraseology and collocation studies. Scholars like Biber et al. (1999), Hunston/Francis (2000), Kjellmer (1984), Lewis (1993, 1997, 2000), Meunier/Gouverneur (2007), Nattinger (1980), Pawley/Syder (1983), and Sinclair/Renouf (1988) have emphasised the importance of recurring word combinations and prefabricated strings in a pedagogical context because of their great potential in fostering fluency, accuracy and idiomaticity. Although corpus-based collocation dictionaries (e. g. Hill/Lewis 1997; Lea 2002) are available, and

although information on phraseology (i.e. about the combinations that individual words favour) is implicitly included in learners dictionaries in the word definitions and the selected corpus examples and sometimes even explicitly described, e. g. in the grammar column in the COBUILD dictionaries and in the *COBUILD Grammar Patterns* reference books (cf. Francis/Hunston/Manning 1996, 1998).

Types of pedagogical corpus applications

When we talk about the application of corpora in language teaching, this includes both the use of corpus *tools*, i. e. the actual text collections and software packages for corpus access, and of corpus *methods*, i. e. the analytic techniques that are used when we work with corpus data. In classifying pedagogical corpus applications, i. e. the use of corpus tools and methods in a language teaching and language learning context, a useful distinction (going back to Leech 1997) can be made between direct and indirect applications. In order to conduct a study of language which is corpus-based, it is necessary to gain access to a corpus and a concordance program.

Uses of Annotated Corpora for Language Teaching and Learning

By using corpus, material developer can create exercises based on the real examples students will get an opportunity to discover features of language use.

Applying corpus linguistics to language teaching

According to Barlow (2002), three realms in which corpus linguistics can be applied to teaching are syllabus design, materials development, and classroom activities.

Syllabus Design

The syllabus organizes the teacher's decisions regarding the focus of a class with respect to the students' needs. Frequency and register information could be quite helpful in course planning choices. By conducting an analysis of a corpus which is relevant to the purpose a particular class, the teacher can determine what language items are linked to the target register.

Materials Development

The development of materials often relies on a developer's intuitive sense of what students need to learn. With the help of a corpus, a materials developer could create exercises based on real examples which provide students with an opportunity to discover features of language use. In this scenario, the materials developer could conduct the analysis or simply use a published corpus study as a reference guide.

Class room activities

These can consist of hands on student-conducted language analyses in which the students use a concordancing program and a deliberately chosen corpus to make their own discoveries about language use. The teacher can guide a predetermined investigation which will lead to predictable results or can have the students do it on their own, leading to less predictable findings. This exemplifies data driven learning, which encourages learner autonomy by training students to draw their own conclusions about language use.

The teacher / students and their role and benefits

The teacher would act as a research facilitator rather than the more traditional imparter of knowledge. The benefit of such student-centered discovery learning is that the students are given access to the facts of authentic language use, which comes from real contexts rather than being constructed for pedagogical purposes, and are challenged to construct generalizations and note patterns of language behavior. Even if this kind of study does not have immediately quantifiable results, studying concordances can make students more aware of language use. Richard Schmidt (1990), a proponent of consciousness-raising, argues that "what language learners become conscious of -- what they pay attention to, what they notice...influences and in some ways determines the outcome of learning." According to Willis (1998), students may be able to determine:

- the potential different meanings and uses of common words
- useful phrases and typical collocations they might use themselves
- the structure and nature of both written and spoken discourse
- that certain language features are more typical of some kinds of text than others

Barlow (1992) suggests that a corpus and concordancer can be used to:

- compare language use--student/native speaker, standard English/scientific English, written/spoken
- analyze the language in books, readers, and course books
- generate exercises and student activities
- analyze usage—when is it appropriate to use obtain rather than get?
- examine word order
- compare similar words--ask vs request

Corpora and reference works and teaching materials

- The results of the above mentioned corpus-course book comparisons do not only inform the language teaching curriculum but also help with decisions about the presentation of items and structures in reference works and teaching materials. Research on general corpora has exerted a huge influence on reference publishing and has led to a new generation of dictionaries and grammar books. Nowadays, “people who have never heard of a corpus are using the products of corpus research.” (McEnery/Xiao/Tono 2006, 97) In the context of ELT (English Language Teaching), the publications in the Collins CO-BUILD series constitute a major achievement. Based on real English and compiled with the needs of the language learner in mind, the COBUILD dictionaries, grammars, usage guides, and concordance samplers (cf. Capel 1993; Carpenter 1993; Goodale 1995; Sinclair et al. 1990; Sinclair et al. 1992; Sinclair et al. 2001) offer teachers and learners more reliable information about the English language than any of the more traditional reference grammars or older non-corpus-based dictionaries.
- Two major advantages of the COBUILD and other corpus-based reference works for learners, e. g. those published in the past few years by Longman, Macmillan, OUP and CUP (cf. e. g. Abbs/Freebairn 2005; Biber/Leech/Conrad 2002; Hornby 2005; Peters 2004; Rundell et al. 2002) are that they incorporate corpus-derived findings on frequency distribution

and register variation, and that they contain genuine instead of invented examples. Particularly worth mentioning here is the student version of the entirely corpus-based *Longman Grammar of Spoken and Written English* (Biber et al. 2002). The importance of presenting learners with authentic language examples has been stressed in a number of publications (cf. de Beaugrande 2001; Firth 1957; Fox 1987; Kennedy 1992; Römer 2004b, 2005a; Sinclair 1991, 1997). Kennedy (1992, 366), for instance, cautions that “invented examples can present a distorted version of typicality or an over-tidy picture of the system”, and Sinclair (1991, 5) calls it an “absurd notion that invented examples can actually represent the language better than real ones”. Thanks to the ‘corpus revolution’, the language learner can today choose from a range of reference works that are thoroughly corpus-based and that offer improved representations of the language she or he wants to study.

- Another branch of general corpora research that has exerted some influence on the design of reference works and, to a lesser extent, teaching materials is the area of phraseology and collocation studies. Scholars like Biber et al. (1999), Hunston/Francis (2000), Kjellmer (1984), Lewis (1993, 1997, 2000), Meunier/Gouverneur (2007), Nattinger (1980), Pawley/Syder (1983), and Sinclair/Renouf (1988) have emphasised the importance of recurring word combinations and prefabricated strings in a pedagogical context because of their great potential in fostering fluency, accuracy and idiomaticity. Although corpus-based collocation dictionaries (e. g. Hill/Lewis 1997; Lea 2002) are available, and although information on phraseology (i.e. about the combinations that individual words favour) is implicitly included in learners dictionaries in the word definitions and the selected corpus examples and sometimes even explicitly described, e. g. in the grammar column in the COBUILD dictionaries and in the *COB UILD Grammar Patterns* reference books (cf. Francis/Hunston/Manning 1996, 1998).

- **Types of pedagogical corpus applications**

When we talk about the application of corpora in language teaching, this includes both the use of corpus *tools*, i. e. the actual text collections and software packages for corpus access, and of corpus *methods*, i. e. the analytic techniques that are used when we work with corpus data. In classifying pedagogical corpus applications, i. e. the use of corpus tools and methods in a language teaching and language learning context, a useful distinction (going back to Leech 1997) can be made between direct and indirect applications. In order to conduct a study of language which is corpus-based, it is necessary to gain access to a corpus and a concordancing program.

- **Uses of Annotated Corpora for Language Teaching and Learning**

By using corpus, material developer can create exercises based on the real examples students will get an opportunity to discover features of language use.

- **Applying corpus linguistics to language teaching**

According to Barlow (2002), three realms in which corpus linguistics can be applied to teaching are syllabus design, materials development, and classroom activities.

- **Syllabus Design**

The syllabus organizes the teacher's decisions regarding the focus of a class with respect to the students' needs. Frequency and register information could be quite helpful in course planning choices. By conducting an analysis of a corpus which is relevant to the purpose a particular class, the teacher can determine what language items are linked to the target register.

- **Materials Development**

The development of materials often relies on a developer's intuitive sense of what students need to learn. With the help of a corpus, a materials developer could create exercises based on real examples which provide students with an opportunity to discover features of language use. In this scenario, the materials developer could conduct the analysis or simply use a published corpus study as a reference guide.

- **Class room activities**

These can consist of hands on student-conducted language analyses in which the students use a concordancing program and a deliberately chosen corpus to make their own discoveries about language use. The teacher can guide a predetermined investigation which will lead to predictable results or can have the students do it on their own, leading to less predictable findings. This exemplifies data driven learning, which encourages learner autonomy by training students to draw their own conclusions about language use.

Computer-Aided language Teaching and learning (CALL)

Computer-aided language learning (CALL), British, or Computer-Aided Instruction (CAI)/Computer-Aided Language Instruction (CALI), American, is briefly defined in a seminal work by Levy (1997: p. 1) as "the search for and study of applications of the computer in language teaching and learning". CALL embraces a wide range of information and communications technology applications and approaches to teaching and learning foreign languages, from the "traditional" drill-and-practice programs that characterised CALL in the 1960s and 1970s to more recent manifestations of CALL, e.g. as used in a virtual learning environment and Web-based distance learning. It also extends to the use of corpora and concordancers, interactive whiteboards, Computer-mediated communication (CMC), language learning in virtual worlds, and mobile-assisted language learning (MALL).

The term CALI (computer-assisted language instruction) was in use before CALL, reflecting its origins as a subset of the general term CAI (computer-assisted instruction). CALI fell out of favour among language teachers, however, as it appeared to imply a teacher-centred approach (instructional), whereas language teachers are more inclined to prefer a student-centred approach, focusing on learning rather than instruction. CALL began to replace CALI in the early 1980s (Davies & Higgins 1982: p. 3) and it is now incorporated into the names of the growing number of professional associations worldwide.

An alternative term, technology-enhanced language learning (TELL), also emerged around the early 1990s: e.g. the TELL Consortium project, University of Hull. The current philosophy of CALL puts a strong emphasis on student-centred materials that allow learners to work on their own. Such materials may be structured or unstructured, but they normally embody two important features: interactive learning and individualised learning. CALL is essentially a tool that helps teachers to facilitate the language learning process. It can be used to reinforce what has already been learned in the classroom or as a remedial tool to help learners who require additional support.

The design of CALL materials generally takes into consideration principles of language pedagogy and methodology, which may be derived from different learning theories (e.g. behaviourist, cognitive, constructivist) and second-language learning theories such as Stephen Krashen's monitor hypothesis.

A combination of face-to-face teaching and CALL is usually referred to as blended learning. Blended learning is designed to increase learning potential and is more commonly found than pure CALL (Pegrum 2009:

A definition of CALL

Computer Assisted Language Learning (CALL) is often perceived, somewhat narrowly, as an approach to language teaching and learning in which the computer is used as an aid to the presentation, reinforcement and assessment of material to be learned, usually including a substantial **interactive** element. Levy (1997:1) defines CALL more succinctly and more broadly as "the search for and study of applications of the computer in language teaching and learning". Levy's definition is in line with the view held by the majority of modern CALL practitioners. For a comprehensive overview of CALL see ICT4LT Module 1.4, *Introduction to Computer Assisted Language Learning (CALL)*: <http://www.ict4lt.org/>.

A brief history of CALL

CALL's origins can be traced back to the 1960s. Up until the late 1970s CALL projects were confined mainly to universities, where computer programs were developed on large mainframe computers. The PLATO project, initiated at the University of Illinois in 1960, is an important landmark in the early development of CALL (Marty 1981). In the late 1970s, the arrival of the **personal computer** (PC) brought computing within the range of a wider audience, resulting in a boom in the development of CALL programs and a flurry of publications. Early CALL favoured an approach that drew heavily on practices associated with **programmed instruction**. This was reflected in the term Computer Assisted Language Instruction (CALI), which originated in the USA and was in common use until the early 1980s, when CALL became the dominant term. There was initially a lack of imagination and skill on the part of programmers, a situation that was rectified to a considerable extent by the publication of an influential seminal work by Higgins & Johns (1984), which contained numerous examples of alternative approaches to CALL. Throughout the 1980s CALL widened its scope, embracing the **communicative approach** and a range of new technologies. CALL has now established itself as an important area of **research** in higher education: see the joint EUROCALL/CALICO/IALLT Research Policy Statement: http://www.eurocall-languages.org/research/research_policy.htm. See also the History of CALL website: <http://www.history-of-call.org/>.

Traditional CALL

Traditional CALL programs presented a **stimulus** to which the learner had to provide a **response**. In early CALL programs the stimulus was in the form of text presented on screen, and the only way in which the learner could respond was by entering an answer at the keyboard. Some programs were very imaginative in the way text was presented, making use of colour to highlight grammatical features (e.g. gender in French and case endings in German) and movement to illustrate points of syntax (e.g. position of

adjectives in French and subordinate clause word order in German). Discrete **error analysis** and **feedback** were a common feature of traditional CALL, and the more sophisticated programs would attempt to analyse the learner's response, pinpoint errors, and branch to help and remedial activities. A typical example of this approach is the *CLEF* package for learners of French, which was developed in the late 1970s and early 1980s by a consortium of Canadian universities. A Windows version of CLEF has recently been released: <http://www.camsoftpartners.co.uk/clef.htm> Error analysis in CALL is, however, a matter of controversy. Practitioners who come into CALL via the disciplines of **computational linguistics**, e.g. Natural Language Processing (NLP) and Human Language Technologies (HLT), tend to be more optimistic about the potential of error analysis by computer than those who come into CALL via language teaching: see ICT4LT Module 3.5, *Human Language Technologies*: <http://www.ict4lt.org/>. The approach adopted by the authors of *CLEF* was to anticipate common errors and build in appropriate feedback. An alternative approach is the use of **Artificial Intelligence** (AI) techniques to **parse** the learner's response - so-called "intelligent CALL" (ICALL) - but there is a gulf between those who favour the use of AI to develop CALL programs (Matthews 1994) and, at the other extreme, those who perceive this approach as a threat to humanity (Last 1989:153).

Explorative CALL

More recent approaches to CALL have favoured a learner-centred, **explorative** approach rather than a teacher-centred, drill-based approach to CALL. The explorative approach is characterised by the use of **concordance programs** in the languages classroom - an approach described as **Data-Driven Learning** (DLL) by Tim Johns (Johns & King 1991). There are a number of concordance programs on the market, e.g. *MonoConc*, *Concordance*, *Wordsmith* and *SCP* - all of which are described in ICT4LT Module 2.4, *Using concordance programs in the modern foreign languages classroom*: <http://www.ict4lt.org/>. See also Tribble & Jones (1990). The explorative approach is widely used today, including the use of **Web concordancers** and other Web-based CALL activities.

Multimedia CALL

Early personal computers were incapable of presenting authentic recordings of the human voice and easily recognizable images, but this limitation was overcome by combining a personal computer and a 12-inch videodisc player, which made it possible to combine sound, photographic-quality still images and video recordings in imaginative presentations - in essence the earliest manifestation of **multimedia CALL**. The result was the development of **interactive videodiscs** for language learners such as *Montevidisco* (Schneider & Bennion 1984), *Expodisc* (Davies 1991), and *A la rencontre de Philippe* (Fuerstenberg 1993), all of which were designed as simulations in which the learner played a key role.

The techniques learned in the 1980s by the developers of interactive videodiscs were adapted for the **multimedia personal computers** (MPCs), which incorporated CD-ROM drives and were in widespread use by the early 1990s. The MPC is now the standard form of personal computer. CD-ROMs were used in the 1980s initially to store large quantities of text and later to store sound, still images and video. By the mid-1990s a wide range of multimedia CD-ROMs for language learners was available, including imaginative simulations such as the *Who is Oscar Lake?* series: <http://www.languagepub.com/>.

The quality of video recordings offered by CD-ROM technology, however, was slow to catch up with that offered by the earlier interactive videodiscs. The **Digital Video Disc** (DVD) offers much higher quality video recordings, e.g. the Eurotalk Advanced Level DVD-ROM series: <http://www.eurotalk.co.uk/>. A feature of many multimedia CALL programs is the role-play activity, in which the learner can record his/her own voice and play it back as part of a continuous dialogue with a native speaker. Other multimedia programs make use of **Automatic Speech Recognition** (ASR) software to diagnose learners' errors, e.g. *Tell Me More Pro* by Auralog: <http://www.auralog.com/english.html>. Most CALL programs under

development today fall into the category of multimedia CALL. See ICT4LT Module 2.2, *Introduction to multimedia CALL*: <http://www.ict4lt.org/>.

Web-based CALL

In 1992 the **World Wide Web** was launched, reaching the general public in 1993. The Web offers enormous potential in language learning and teaching, but it has some way to go before it catches up with the interactivity and speed of access offered by CD-ROMs or DVDs, especially when accessing sound and video files. For this reason, Felix (2001:190) advises adopting **hybrid** approaches to CALL, integrating CD-ROMs and the Web and running audio conferencing and video conferencing in conjunction with Web activities. The Web Enhanced Language Learning (WELL) project, which has been funded under the FDTL programme of the HEFCE, aims to promote wider awareness and more effective use of the Web for teaching modern languages across higher education in the UK. The WELL website provides access to high-quality Web resources in a number of different languages, selected and described by subject experts, plus information and examples on how to use them for teaching and learning: <http://www.well.ac.uk/>.

See also the following ICT4LT modules: <http://www.ict4lt.org/>

1.5 *Introduction to the Internet*

2.3 *Exploiting World Wide Web resources online and offline*

3.2 *Creating a World Wide Web site*

CALL authoring programs

CALL authoring programs offer a do-it-yourself approach to CALL. They were originally developed to enable programmers to simplify the entry of data provided by language teachers. Modern CALL authoring programs are designed to be used by language teachers who have no knowledge of computer programming. Typical examples are authoring packages that automatically generate a set of pre-set activities for the learner, e.g. Camsoft's *Fun with Texts* (Camsoft) and *The Authoring Suite* (Wida Software). Generic packages such as Macromedia's *Director* (<http://www.macromedia.com/>) are more sophisticated and

enable the user to create a full-blown course, but they are probably too complex for most language teachers and are best suited to the **template approach** to authoring, as described in ICT4LT Module 3.2, *CALL software design and implementation*: <http://www.ict4lt.org/> **Web authoring packages** are also available, e.g. *Hot Potatoes* software: <http://web.uvic.ca/hrd/halfbaked>. See ICT4LT Module 2.5, *Introduction to CALL authoring programs*. See also Bickerton (1999) and Bickerton, Stenton & Temmermann (2001).

Professional associations for CALL

An increasing number of professional associations devoted to CALL are emerging worldwide. The older associations are grouped together under WorldCALL, which is in the process of establishing itself as an umbrella association of associations. WorldCALL held its first conference at the University of Melbourne in 1998, and the second WorldCALL conference will take place in Banff, Canada, 2003: <http://www.worldcall.org/>. The current professional associations represented in WorldCALL are:

EUROCALL: The leading European professional association for CALL. The *ReCALL* journal is published by Cambridge University Press on behalf of EUROCALL: <http://www.eurocall-languages.org>

CERCLES: The European Confederation of Language Centres in Higher Education. <http://www.cercles.org/>. CERCLES embraces a similar constituency to IALLT in North America.

CALICO: The leading North American professional association for CALL. Publishes the *CALICO Journal*: <http://www.calico.org/>

IALLT: International Association for Language Learning Technology, based in North America: <http://www.iallt.org/>. IALLT publishes the *IALLT Journal of Language Learning Technologies* and embraces a similar constituency to CERCLES in Europe.

CCALL/ACELAO: Currently in the process of establishing itself as a formal professional association in Canada. No website is available at present.

LLA: The Language Laboratory Association of Japan, also known as LET, which now embraces a wider range of language learning technologies: <http://langue.hyper.chubu.ac.jp/lla>

ATELL: The Australian Association for Technology Enhanced Language Learning consortium: <http://www.arts.uq.edu.au/ATELL>. ATELL used to publish *On-CALL*, which has now merged with *CALL-EJ* (Japan).

CALL is our main area of research and interest within the field of Applied Linguistics and Language Teaching. Below you will find some descriptions of key concepts in the field, as well as links to other sites which provide useful links.

- **CALL** : Computer Aided Language Learning – this is the most commonly used acronym to encompass all elements of language learning that takes place with or around computers. It is the general term and is often thought to incorporate other terms such as CMC, TELL, CASLA and so on. Note that although many people accept CALL as the umbrella term, there are also important distinctions between CALL and the other concepts. *See Levy & Hubbard ‘Why call CALL ‘CALL’?’ in Computer Aided Language Learning*
- **CMC** : Computer Mediated Communication – This refers to any form of communication which uses computers as the medium. For example, forums, chat rooms, texts and SMS. Emails and VOIP conversations would also fall under this category. As CMC advances, more and more of CMC comes to simulate face-to-face communication. The use of WebCams and VOIP have meant that the gap between CMC and face-to-face is ever growing smaller and the varieties of CMC are expanding.

- **CASLA:** Computer Applications in Second Language Acquisition – Carol Chapelle (2001) from Iowa State university wrote a book defining CASLA and presents a useful framework for analysing and evaluating them. These principles are often used to analyse CALL software and content. Chapelle uses CASLA as an umbrella term to encompass CALL and Computer Assisted Language Testing (CALT) Computer Adaptive Testing (CAT) and talks of other related fields such as Educational Technology, Computer Supported Collaborative Learning (CSCL) and Artificial Intelligence (AI).
- **TELL:** Technology Enhanced Language Learning – This would encompass approaches that utilise the role of computers and any other devices that fall under the definition of ‘Technology,’ for example MP3s, Podcasts, Video, DVD, mobile applications, GPS, Remote Access Field Trips (RAFT), social networking sites, Second Life, Virtual Learning Environments and so on. The difference here being that TELL is likely to still feature within a more traditional classroom setting with the addition of a host of tools such as the above being integrated so as to improve the opportunities for meaningful and authentic communication and language use.
- **ICT:** Information Communication Technology – Like IT (Information Technology) but also encompassing telephones, VOIP, and any other technology that allows for human communication and access to information. This is not a specific term for language learning and teaching.