Iman Tohidian
University of Kashan
Kashan, Iran

Corpus-based Translation Method

Abstract. Machine Translation (MT) is the oldest application area of computational linguistics, dating back to the early years of the Cold War, and MT is one of the biggest application areas for computational linguistics, with technical manuals, office materials, and other communications being translated daily. Unlike the translation of literary texts, where a considerable amount of creativity is required on the part of the translator, Machine Translation is focused on translations which preserve the information content of the source language as much as possible, while rendering it in a natural form in the target language. Its main advantages are economic, particularly when the volume of text is such that humans could not possibly translate it. Lower accuracy translations may be sufficient for getting the gist of some foreign language source, whereas for higher-quality results, post-editing of the machine translation by humans is often necessary. For my research in finding a new way to help those who use MT and solve these problems, I found a link between theories which may be used to achieve better machine translation. To do so, I used *Corpus Linguistics* along with two other theories called, *Computational Linguistics* and *Artificial Intelligence.*

Key Words: Annotation; colligation; collocation; compile; concordance; context; corpus; encoding; hit; KWIC (Key-Word In Context); lemma; mark-up; match; natural language; NLP (Natural Language Processing); parsing; POS (Part of Speech); part of speech tagging; SGML (Standard Generalized Mark-up Language); string; tag; TEI (Text Encoding Initiative); token; Treebank; type.

Introduction

The word *Corpus,* derived from the Latin word meaning *body,* may be used to refer to any text in written or spoken form. In modern linguistics, this term is used to refer to large collections of texts represent a sample of a particular variety or use of languages that are presented in machine readable form. There are many levels of information that can be gathered from a corpus. These levels range from simple word lists to catalogues of complex grammatical structures and interactive analyses that can reveal both linguistic and nonlinguistic association patterns. Analyses can explore individual lexical or linguistic features across texts or identity clusters of features that characterize particular registers (Biber, 1988). The tools that are used for these analyses range from basic concordancing packages to complex interactive computer programs.

Corpus linguistics is the study of language as expressed in samples (corpora) or "real world" text. This method represents a digestive approach to deriving a set of abstract rules by which a natural language is governed or else relates to another language. Originally done by hand, corpora are largely derived by an automated process, which is corrected. The core of a corpus is the derivation of a set of Part-of-speech tags, representing a formal overview of the various types of words and word-relationships in a given language.

Computational methods had once been viewed as a holy grail of linguistic research, which would ultimately manifest a rule set for natural language processing and machine translation at a high level. The importance of corpora to language study is aligned to the importance of empirical data. Empirical data enable the linguist to make objective statements, rather than those which are subjective, or based upon the individual's own internalized cognitive perception of language. Empirical data also allows us to study language varieties such as dialects or earlier periods in a language for which it is not possible to carry out a rationalist approach.

The corpus approach is opposite to Noam Chomsky's view that real language is puzzled with performance-related errors, thus requiring careful analysis of small speech samples obtained in a highly controlled laboratory setting. Corpus linguistics does away with Chomsky's competence/performance split; adherents believe that reliable language analysis best occurs on field-collected samples, in natural contexts and with minimal experimental interference. Corpus linguistics should be seen as a subset of the activity within an empirical approach to linguistics. Although corpus linguistics entails an empirical approach, empirical linguistics does not always entail the use of a corpus.

Layout

The first or most basic information that we can get from a corpus is frequency of occurrence information. A word list is simply a list of all the words that occur in the corpus. These lists can be arranged in alphabetic or frequency order. Word lists derived from corpora can be useful for vocabulary instruction and test development. In addition to frequency lists, concordancing packages can provide additional information about lexical co-occurrence patterns. To generate a concordance listing showing these patterns, a target word or phrase needs to be selected. Once the search word/ phrase is selected, the program can search the texts in the corpus and provide a list of each occurrence of the target word in context. A concordance program can also provide information about words that tend to occur together in the corpus. For example, we could discover which words most frequently occur just to the right or to the left of a particular target word, or even within two or three words to the left or right of the target word. Words that commonly occur with or in the vicinity of a target word are called collocates, and the resulting sequences or sets of words are called collocations. For example, the nearly synonymous verbs *begin* and *start* have the same grammatical potential. That is, they can be used with the same variety of clause elements (for example, transitive, intransitive).

Text Encoding and Annotation

If corpora is said to be *unannotated* it appears in its existing raw state of plain text, whereas *annotated* corpora has been enhanced with various types of linguistic information. Unsurprisingly, the utility of the corpus is increased when it has been annotated, making it no longer a body of text where linguistic information is *implicitly* present, but one which may be considered a repository of linguistic information. The implicit information has been made explicit through the process of concrete annotation.

For example, the form "gives" contains the implicit part-of-speech information "third person singular present tense verb" but it is only retrieved in normal reading by 'recourse to our pre-existing knowledge of the grammar of English. Such annotation makes it quicker and easier to retrieve and analyze information about the language contained in the corpus.

As all of us know, morphology is the study of the structure of words, so the task of a machine is to take a word in a language and break it down into its stem form along with any suffixes and prefixes that it may have attached to that stem. In analyzing a sentence such as *Ali sang well,* the machine should be able to identify *Ali* as a proper name, *sang* as the irregular past form of the verb *sing,* and *well* as an adverb. Here we should know that the machine will not be able to identify the syntactic roles of words, for example *Ali* is the subject of *sang,* and this is the subsequent task of a syntactic parsing program. An important strategy in computational linguistics is to treat language as modular or composed of different subsystems and to develop and integrate modules for different subsystems.

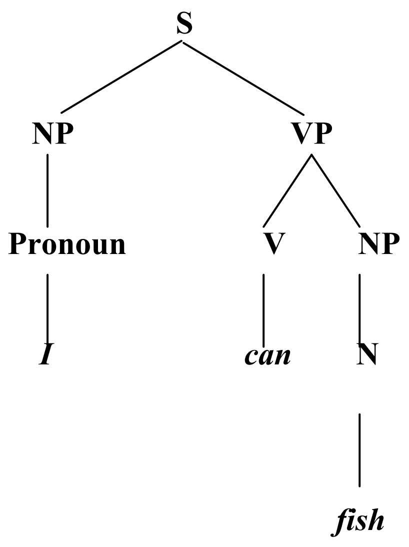
The first step in the morphological analysis is to identify separate words, which is called *tokenization.* It's too easy in languages like English, where words are delimited by spaces and punctuation characters, and where sentences start with capital letters. But even in English, ambiguous punctuation can cause tokenization problems. For example, periods may be part of an abbreviation (*USA products*) or in Persian language, abbreviations such as ایسنا ‘ISNA’ (Iranian Student's News Agency) or ایرنا ‘IRNA’ (Iranian News Agency).

A machine translator takes a word and breaks it down into its components. Sometimes, instead of a full morphological analysis, a simple stemming algorithm is used which deletes suffixes to arrive at a stem form. Another strategy is to use a fully inflected lexicon, which includes all the possible affixed forms of every word in the language. The machine looks up the word in the list, but those lists are almost inevitably incomplete and they can become too large for computers to handle. Language learners are familiar with books that teach a rule (a regular pattern in the target language like "to form the past tense of a verb in English, add –*ed*") and then teach all the exceptions to the rule (irregular verbs like *bring* and *come).* One approach to a full morphological analysis is to store only exceptional forms in the lexicon and handle the regular patterns with morphological rules. Pattern-action approach is a computer program which identifies words that match the rule's pattern and then it records that word's morphological components as specified by the rule's action.

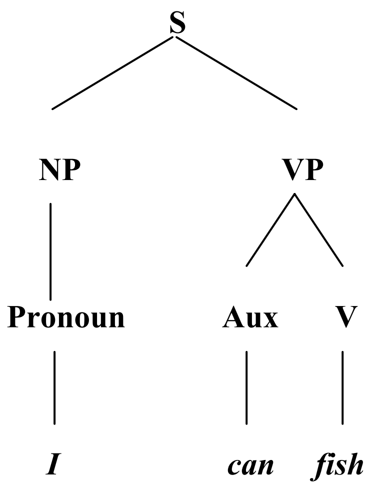
Now I want to introduce the problem of trying to get a machine to characterize the grammatical structure of a sentence. Given a set of linguistic rules that describe how elements of a sentence can be put together, a computer program called a syntactic parser will try to find the best grammatical analysis of a sentence. If the sentence is ambiguous, I mean if it has more than one possible grammatical structure, the syntactic parser will produce all analyses. Consider this short sentence:
I can fish.

This sentence could mean that I know how to fish, or that I habitually put fish in cans. In the first reading, *can* is a modal auxiliary verb,, but in the second sentence, *can* is the main verb. These two different meanings correspond to distinct syntactic structures. We can describe the structure of sentences like the above mentioned in terms of a grammar, expressed as a system of rules. These rules called phrase structure rules, break up a sentence into its constituent parts, consisting of syntactic phrase or words. A computer program which analyzes the syntactic structure of a sentence like *I can fish,* is called a parser. A parser takes an input sentence and produces one or more syntactic representation of it. It produces a single representation if the sentence is syntactically unambiguous, but more than one representation if there is syntactic ambiguity as *I can fish.* One way to represent the hierarchical syntactic structure of a sentence is called a parser tree. Here are the two parser tree for *I can fish,* the parser takes the sentence as input and processes it using the grammar to produce parser tree (a) and as output (b).

(a)



(b)



As we have seen, a word (like *can* and *fish*) can have different possible parts of speech. Instead of having the parser consider all parts of speech of an ambiguous word, it is possible to reduce the ambiguity prior to parsing by running a program called a part of speech tagger before parsing. To each word that has more than one part of speech, the tagger assigns the most likely part of speech. This based on context-based rules derived by human intuition, for example after *the, fish* is likely to be a noun, as well as rules derived by machines that learn from a collection of example sentences that have been tagged already with parts of speech. One problem with context free grammars is that they require many rules. For example, if we want to add subject-verb agreement to the grammar, so that we get a successful parse for *I read* and *he reads,* we would have to create two S rules, one which combines third person singular NP with third person singular VP, and another which combines Non-third person singular NP with Non-third person singular VP. In addition we would have to create separate rewrite rules for each of these more specific NP and VP categories.

So far we have discussed linguistic rules which have been designed by hand by linguists. It's also possible to train a machine to discover the rules from examples of linguistic analyses. For example, in the case of syntax, the linguist may provide only the parse tree, from which the computer can discover the grammar by studying examples of parse trees. To do this, linguists need to decide on the phrase structure rules that will be included in the grammar. The linguists then take a collection of texts called a corpus, and analyze and mark up the sentences in those texts with parse trees. Once a large corpus has been annotated in this way, creating a Treebank, computers can be taught to include grammars from it.

Now I want to show how a computer can learn a grammar from a treebank. First, a program counts how often each type of syntactic configuration is found in the annotated treebank. Each configuration is represented as a rule like the ones in our toy grammar and each rule is weighted according to how often that configuration appears in the treebank. These weighted rules would then be used in statistical parsing and then would search for the parse tree with the highest probability. The probability of a parse tree can be viewed as the probability of all the rules in that parse tree occurring together.

The need for human mark-up make the construction of annotated corpora an expensive capital investment, of course, but once the statistical parser has been developed, it can very quickly parse an indefinite number of new sentences in the language with a measurable level of accuracy. Accuracy is measured by selecting test sentences and comparing the constituents in the treebank's parse tree. Grammars included by machines tend to perform at least as well as grammars developed by hand based on human intuitions. Statistical parsers trained on context-free grammar parse trees from a treebank achieve about 90 percent accuracy on various test sets.

Assume that we have carried out syntactic parsing of the utterance, and we have obtained a parse tree for it. If we start with the parse tree, we can associate meanings with the words at the bottom of the tree, and use that information and the structure of the tree to provide a meaning for the sentence. For a syntactically unambiguous sentence, consisting of words which each have only one meaning, this would be relatively simple. Each word at the bottom of the tree would be looked up in a lexical database; especially a digital dictionary and the meanings of words would be connected together according to the structural relations represented in the parse tree, to produce the meaning of the sentence. Words can be lexically ambiguous. Not only can a word have more than one part of speech, but even a word which is a given part of speech may have more than one meaning. For example, even if we leave out its several meanings as a verb, the noun *spot* could mean a particular location (*we found a nice spot for lunch*) or a stain (*out, out, dammed spot*). A computer uses a program called a word-sense disambiguator to decide which meaning is intended. A word-sense disambiguator can use the context of neighboring words in the sentence as well as other words in the document to figure out which meaning of a given word is most likely. Like a part of speech tagger, it uses rules that depend on context, and these rules can be derived by human intuition or by training a machine learning program.

Now that we have an idea of how to capture word meaning in a computer, we can turn to the problem of constructing a meaning for a sentence based on the meanings of the words in it. A common assumption underlying computational approaches to semantics is that the meaning of the whole is systematically composed of the meaning of the parts. A computer program based on computational semantics puts together the sentence meaning from the meanings of the words and phrases that compose it.

A simple corpus could consist of raw texts, with no additional information provided about the origins, authors, speakers, structure or contents of the texts themselves. However, encoding some of this information in the form of markup makes the corpus much richer and more useful, especially to researchers who were not involved in its compilation. Structural markup refers to the use of codes in the texts to identify structural features of the text. For example, in a written corpus, it may be desirable to identify and code structural entities such as titles, authors, paragraphs, subheadings, chapters, etc. In a spoken corpus, turns and speakers are almost always identified and coded, but there are a number of other features that may be encoded as well, including, for example, contextual events or paralinguistic features. In addition to structural markup, many corpora provide information about the contents and creation of each text in what is called a header attached to the beginning of the text, or else stored in a separate database. Information that may be encoded in the header includes, for spoken corpora, demographic information about the speakers such as age, social class, when and where the speech event took place, relationships among the participants and so forth. For written corpora, demographic information about the author(s), as well as title and publication details may be encoded in a header. For both spoken and written corpora, headers sometimes include classifications of the text into categories, such as register, genre, topic domain, discourse mode or formality.

Types of corpora:

We can say that types of corpora and research topics are the two issues which are related to each other. To put it in simple words, we can say that there are many types of corpora as there are research topics in linguistics. There are a number of different corpora used by researchers in the world.

Corpora can be classified into two types: 1) general corpora 2) specialized corpora.

General corpora:

1) The Brown, the LOB corpus and the BNC are examples of the general corpora.

2) A general corpus is designed to include language samples from a wide range of registers or genres.

3) Most of the early general corpora were limited to written language but many of the modern general corpora include a spoken component.

Specialized corpora:

1) Specialized corpora may include both spoken and written components.

2) The International Corpus of English and the TOEFL -2000 spoken and written Academic Language Corpus are the two examples of specialized corpora.

3) A specialized corpus focuses on a particular spoken or written variety of language.

4) Another type of specialized corpus is learner's corpus. It includes spoken or written language samples produced by non-native speakers.

Corpus Design:

This part talks about the importance of corpus design and the issues which must be considered in designing the corpus. Corpus design can influence the reliability of the results and it can affect the analysis. One of the features of a well-designed corpus is that it should be representative of the types of language included in it**.** Register analysis, topics, speaker type should be considered here. In designing the corpus, the research goals should also be paid attention to**.**

Corpus Compilation:

To create a corpus, we should collect data and create electronic versions of the texts. To do so, we need a scanner and OCR Software. Sometimes, we need to type the materials. To create a corpus, we can use a number of sources on the Internet but the point is that we can not exclusively rely on electronic produced texts. Building a spoken corpus takes a lot of time and needs much money. The transcription system should be decided upon in the first step. Another point to be considered is that how the interactional characteristics of the speech are represented in the transcripts.

Conclusion:

As we have seen, a variety of methods for getting computers to process human languages. The field is motivated in part by the fundamental relationship between patterns found in human languages and mathematical models of systems that process artificial languages. Linguistic theories go hand-in-hand with computational representations and algorithms to address natural language problems. The success of computational linguistics is due in large part to the availability of large quantities of online data that can be processed very quickly. This has led to the use of statistical approaches, displacing some of the earlier approaches that involved hand-created models of linguistic usage. The success of these statistical approaches poses fundamental questions for many current linguistic theories. While computer algorithms and statistical models are very well understood, getting computers to acquire the vast amounts o linguistic and world knowledge needed remains a major challenge. Using this theory will help us to have the best dictionaries nearly for all the languages in the world and getting the best translation to the target language in accordance with the rules of the source languages.

References

Boden, M.A. (1987). *Artificial Intelligence and Natural Man.* Basic Books, New York.

Clarke, D.F. (1986). *Computer-assisted reading – What Can the Machine Really Contribute? Systems 14* 1-13

Kenning, M.J. and Kenning, M. M. (1990). *Computers and Language Learning: current theory and practice.* Ellis Horwood.

O'Connor, J. (2001). *NLP Workbook.* HarperCollins Publishers Limited.

Contact information

Iman Tohidian
Department of English, Faculty of Humanities
University of Kashan, Kashan, I. R. Iran
E-mail: tohid_483@yahoo.com