Multilingual Grammars (for Translation)

Lavanya Aprameya

CS 4744, Spring 2017

This write-up describes a system for translation using multilingual grammars (MGT). The write-up will first outline the motivation for such a system, followed by a description of theories and approaches used to produce the system, culminating in the results of the system, a reflection on the process and the system itself, and a list of future extensions.

Motivation

In general, publicly available computational systems made for translation lack the robustness to represent the morphosyntactic properties of the phrases involved in translation, both the phrases from which and to which translation occurs. This disregard for underlying linguistic representations is made apparent by cases such as demonstrated by (1), (2), and (3). (1) represents the input into a commonly used translation system, Google Translate, in English. (2) represents the result of translating (1) into Spanish, and (3) represents the result of translating (2) into English once again.

(1) I feel like Google Translate can't understand more sophisticated linguistics.

(2) Siento que Google Translate no puede entender una lingüística más sofisticada.

(3) I feel that Google Translate can not understand more sophisticated linguistics.

Two important features of (3) in comparison to (1) to note are that "like" is translated into "that" and that "can't" is translated into "can not". The first change is not necessarily significant, since the words "like" and "that" can be used interchangeably in this construction, but the second change is more significant since it introduces ambiguity that is not present in (1). "can't" in (1) is interpreted to mean that Google Translate does not have the ability to understand, while "can not" (3) can have an additional interpretation: that Google Translate has the ability not to understand. This distinction, which an underlying syntactic representation can clarify, is not an issue with which translational systems typically concern themselves. However, several flaws in such systems can be resolved by taking morphosyntax into account. MGT hopes to avoid such mistakes by using an underlying representation rooted in morphosyntactic theory in order to translate between languages.

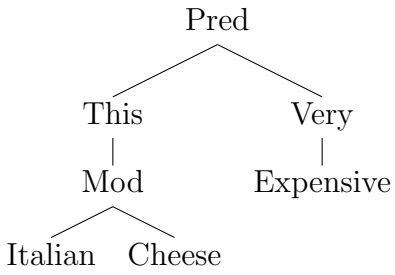
Theories & Approaches

The main theory in linguistics which drives MGT is the idea that multiple languages can be represented by a similar layout, which is explored in Aarne Ranta's *Grammatical Framework* [1]. The computational linguistic theories of unification and feature structures [2] [3] provide the groundwork for MGT, which is finally realized as a 7-step process.

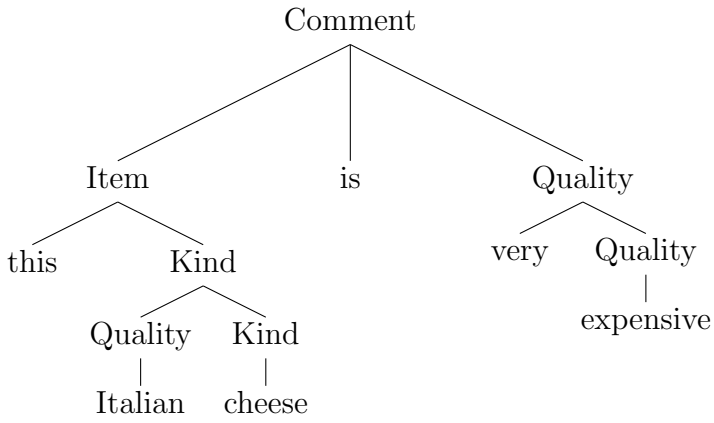
Grammatical Framework

In *Grammatical Framework*, Ranta proposes the idea that languages contain some representation which is inherent in all languages, but the manner in which this representation is realized can differ between languages. For example, the English sentence "This Italian cheese is very expensive" might be represented by a structure such as the following:

(4) (a)



(b)



(4a) represents what Ranta refers to as the *abstract tree* of the sentence, while (4b) represents the *parse tree*. An abstract tree can be turned into a parse tree through a series of linearization rules, and in the same way that abstract trees can be manipulated into parse trees, parse trees can also be manipulated into abstract trees. Since all languages translate the same abstract tree through variations of linearization rules, the parse tree for the target language of a translation can be produced by manipulating the parse tree of the source language in order to match the target language's linearization rules. Using this concept of an underlying abstract representation which allows for such manipulation, MGT concerns itself with the process of manipulating from parse tree to parse tree.

Unification Grammars & Feature Structures

In order to translate between languages, certain features of sentences which are either independent of or not guaranteed by syntactic categories must be taken into account. This core idea which drives unification grammars and feature structures proves to be essential to the MGT framework.

Features are aspects of either a lexical entry or syntactic category which mark something particular. For example, a noun in English might have the feature NUMBER, which is realized as either SINGULAR or PLURAL and represents content of the noun which would not have been apparent through simply its syntactic category. A *feature structure* such as (5) can be used to represent any noun that is masculine and third person singular. In unification

grammars, a syntactic tree contains nodes of such feature structures rather than strings.

(5) $\begin{bmatrix} \text{Gender} & \text{M} \\ \text{Number} & \text{Sing} \\ \text{Person} & 3 \end{bmatrix}$

As the name indicates, unification grammars are grammars in which feature structures *unify* in order to produce the correct final feature structure. For example, the subject NP of the sentence and the verb must agree in person and number. In unification grammars, the features of both of these nodes are compared, and if they do not clash, the sentence takes on the unified feature structure. If unification results in a clash, the sentence is ungrammatical. Feature structures might be underspecified for one or more features in some nodes, in which case unification with another node which specifies for those features results in a feature structure with the most specific feature. The grammar itself specifies for which features unify under what circumstances. (6) outlines a simple example of unification with the syntactic rule NP $\rightarrow$ Det N.

(6) $\begin{bmatrix} \text{Entry} & \text{the} \\ \text{Category} & \text{Det} \\ \text{Person} & 3 \end{bmatrix} + \begin{bmatrix} \text{Entry} & \text{dog} \\ \text{Category} & \text{N} \\ \text{Number} & \text{Sing} \\ \text{Person} & 3 \end{bmatrix} = \begin{bmatrix} \text{Entry} & \text{the dog} \\ \text{Category} & \text{NP} \\ \text{Number} & \text{Sing} \\ \text{Person} & 3 \end{bmatrix}$

The vocabulary in unification grammars consists of words which map to feature structures, with the most correct and most specific feature structure being the one that takes on the mapping in parsing and translation. Unification grammars are necessary for multilingual grammars since not all languages specify for the same features. For example, English does not specify for case, but German does, and in order to translate between the languages, the feature (case) is overspecified in some cases and sufficiently specified in others, but since vocabulary selection chooses the most correct and most specific, overspecification is not an issue for translation.

MGT Framework

The general framework of MGT is the following:

STEP 1 The user chooses the source and target languages.

STEP 2 The user inputs the phrase to be translated.

STEP 3 MGT parses the input using a binary chart-parsing algorithm [4]. If parsing fails, none of the next steps occur.

STEP 4 MGT imposes the feature structures specified by the grammar onto the parse tree.

STEP 5 MGT performs unification on the parse tree. If unification fails, none of the next steps occur.

STEP 6 MGT finds the corresponding feature structure for each node in the target language and manipulates the parse tree accordingly.

STEP 7 MGT outputs both the resulting parse tree and string in the target language.

Each of these steps are described further below.

Step 1: Choosing the source and target languages

The choice of language pairs lies in the choice of function from the file `translate.ml`. Available functions include `english_to_german`, `english_to_italian`, `german_to_english`, `german_to_italian`, `italian_to_english`, and `italian_to_german`.

Step 2: Inputting the phrase to be translated

The argument to each of the above functions is a string, which each function breaks up into a list of words to be parsed through. The string can represent any sequence, but only sequences recognized by the grammar will produce an output. Any phrase which has a rule in the grammar is allowed, meaning that MGT translates not only sentences, but also stand-alone noun phrases, verb phrases, etc.

Step 3: Chart-parsing

The chart-parsing algorithm which MGT utilizes should be attributed to John Hale, who made his code available to his CS 4744, Spring 2017 class. This parser uses a chart in order to memoize parsed categories and to allow for syntactically ambiguous structures to be represented in all possible ways specified by the grammar. The grammar is binary and allows for intransitive verbs, transitive verbs, and verbs which subcategorize for subordinate clauses. While the grammar is extended in MGT to include features, the chart-parser ignores features and focuses solely on the word entries, while features play a role in the next steps.

Step 4: Imposing feature structures

Each rule within the grammar is of the form FEATURE-STRUCTURE → FEATURE-STRUCTURE FEATURE-STRUCTURE, with feature structures bundling together the features PERSON, NUMBER, GENDER, CASE, and SUBORDINATE. Feature structures also contain a LEX to represent the meaning of the node as well as a ENTRY to represent the realization of the node. The feature structures corresponding to each rule are added into the parse tree at this step.

Step 5: Unifying feature structures

Each feature can manifest itself in the grammar as one of `Any`, `Only(s)`, `Unifier(s)`, and `UnifyBlock(f)`, where `Any` represents a feature which is not specified for, `Only(s)` represents a feature which specifies `s`, `Unifier(s)` represents a feature which specifies `s` and is used in the unification process, and `UnifyBlock(f)` represents a feature which has the feature value `f` and serves as a block for higher unification. Unification functions as described before and continues higher up into the tree until it reaches the final node or a `UnifyBlock`. If any of the unifications result in a clash, the feature in question manifests itself as `Clash`. At the end of the unification process, all parse trees containing `Clash` are disallowed from translation.

Step 6: Manipulating the parse tree into the target language

Using the idea that languages have the same underlying representation, MGT uses grammars which have rules for the same categories across languages. MGT begins to analyze the unified parse tree from the topmost node, searching for each rule in the target language and either keeping the branches the same or reordering them based on what rule the target language maps the node to. Terminal nodes also search for the LEX feature of the original terminal node in order to map to the word with the correct meaning. German undergoes other manipulation once this base manipulation has occurred. Since it is an underlyingly SOV language but undergoes V2 movement in non-subordinate clauses, languages translating to German must undergo this movement before finalizing the parse tree. However, German specifies for both SOV and SVO word-ordering with different values of SUBORDINATE. Therefore, only trees in which the topmost node is non-subordinate are allowed to translate into other languages.

Step 7: Outputting the parse tree and string of the target language

MGT turns the manipulated parse tree into a PDF with the name ⟨`function name`⟩ ⟨ `index of parse tree in list of valid parsetrees`⟩`.pdf`. It also uses a `to_string` function to map the tree to a string of the correct order.

Results of MGT

MGT successfully parses for sentences which include intransitive, transitive, and subordinate clause-specifying verbs, as well as phrases which requires unification, across English, Italian, and German.

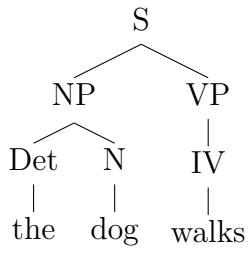
The sentences in (7)-(9) represent one set of intransitive sentences across English, Italian, and German, with trees (10)-(12) as the output of MGT for these sentences from either of the other two languages. The feature structures have been left out of the output of the trees for readability.

(7) the dog walks

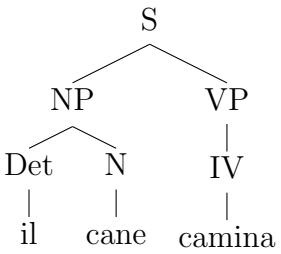
(8) il cane cammina

(9) der hund geht

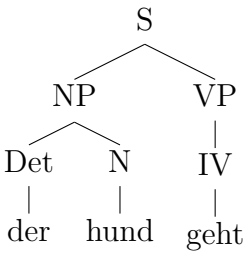
(10)



(11)



(12)



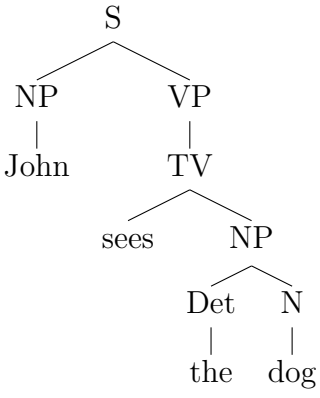
The sentences in (13)-(15) represent one set of transitive sentences across English, Italian, and German, with trees (16)-(18) as the output of MGT for these sentences from either of the other two languages. The feature structures have been left out of the output of the trees for readability.

(13) John sees the dog

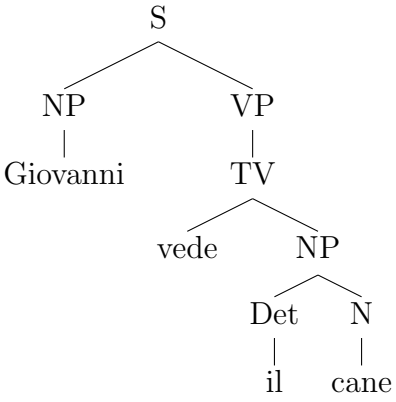
(14) Giovanni vede il cane

(15) Johannes sieht den hund

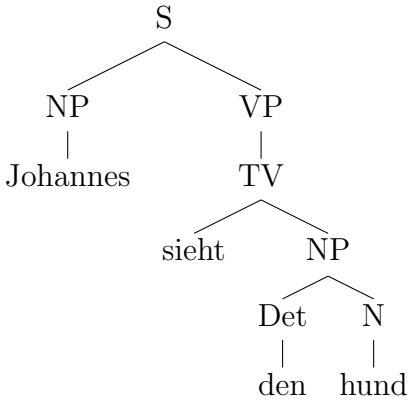
(16)



(17)



(18)



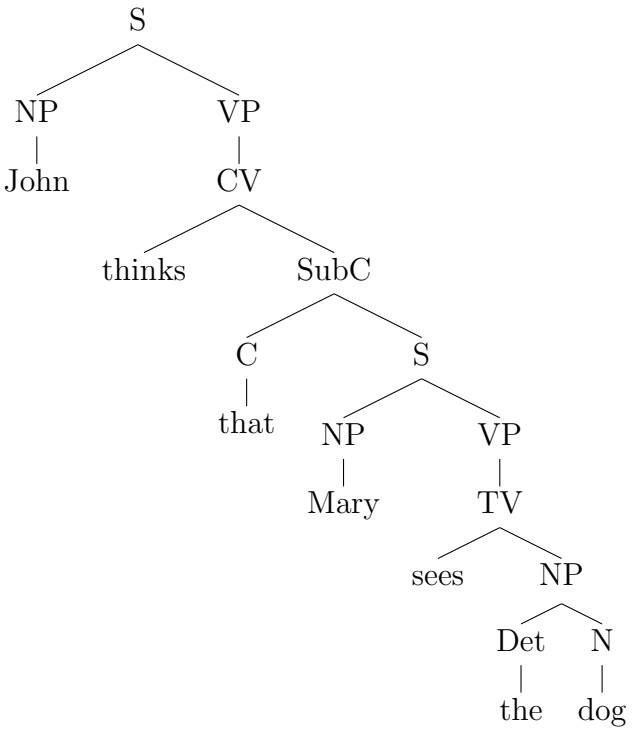
The sentences in (19)-(21) represent one set of transitive sentences across English, Italian, and German, with trees (22)-(24) as the output of MGT for these sentences from either of the other two languages. The feature structures have been left out of the output of the trees for readability.

(19) John thinks that Mary sees the dog

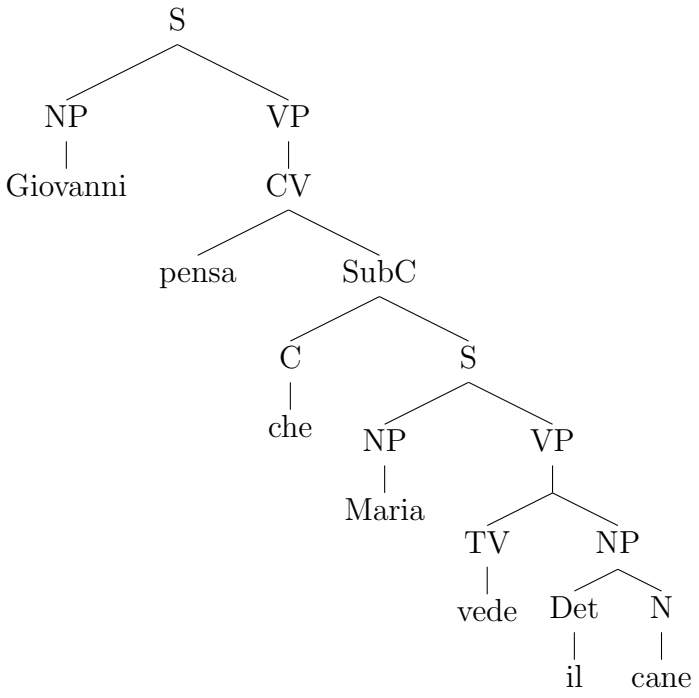
(20) Giovanni pensa che Maria vede il cane

(21) Johannes denkt dass Maria den hund sieht

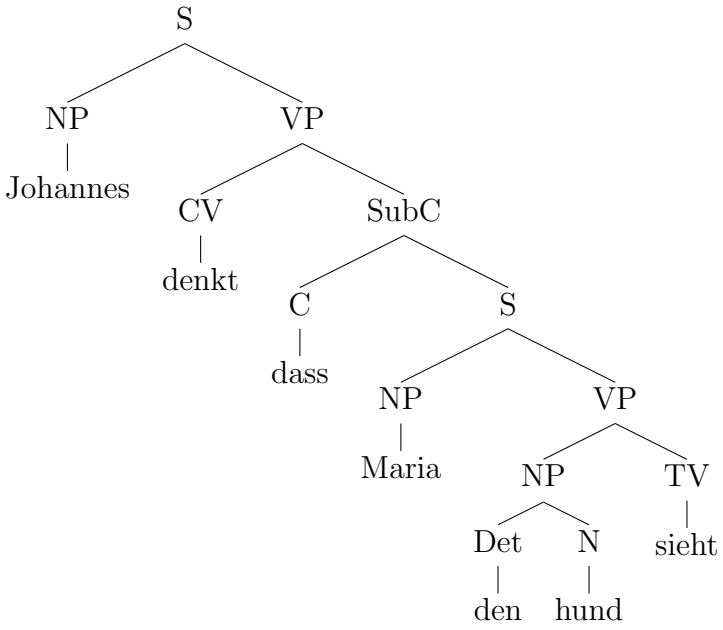
(22)



(23)


S
NP
VP
Giovanni
CV
pensa
SubC
C
S
che
NP
VP
Maria
TV
NP
vede
Det
N
il
cane

(24)


S
NP
VP
Johannes
CV
SubC
denkt
C
S
dass
NP
VP
Maria
NP
TV
Det
N
sieht
den
hund

Reflections & Future Extensions

While MGT's success at representing an underlying feature structure across languages and utilizing it for translation was satisfying, the process itself was more tedious than expected, especially in the way in which German V2 movement was represented. The original goal was to represent V2 movement solely through two different rules in German: VP[+SUBORDINATE] → NP TV and VP[-SUBORDINATE] → TV NP. While this parses correctly in German, translation always chooses the underlying rule due to underspecification in other languages and rule ordering. Therefore, the ways in which MGT now accounts for this hole seem to be not as satisfactory as possible. Another problem which arose is in non-binary grammars and imposing features, since non-binary grammars eliminate intermediate categories (i.e., TV → IV and IV → walks are realized simply as TV → walks). This category elimination results in a loss of essential feature structure information. The manner in which MGT approaches unification also seems to be controversial: in other unification grammars, there is no distinction between `Unifier(s)` and `Only(s)`, and there is no need for `UnifyBlock(f)` since unification applies on a rule-by-rule basis. While MGT's system is efficient, it perhaps does not represent the linguistic theories behind unification in the way that it should.

Therefore, some future extensions include incorporating V2 movement into rules without having to disallow certain parses, as well as incorporating non-binary grammars. Finally, the most important and constructive extension is to extend MGT to a language which is rather dissimilar to the current languages which MGT recognizes in order to test Ranta's hypothesis of multilingual grammars.

References

[1] Ranta, Aarne. *Grammatical framework: Programming with multilingual grammars*. CSLI Publications, Center for the Study of Language and Information, 2011.

[2] Farghaly, Ali Ahmed Sabry. *Handbook for language engineers*. CSLI Publications Stanford University, 2003.

[3] Bresnan, Joan and Asudeh, Ash and Toivonen, Ida and Wechsler, Stephen. *Lexical-functional syntax*. John Wiley & Sons, Vol. 16, 2015.

[4] Mellish, CS and Whitelock, P and Ritchie, GD. *Techniques in Natural Language Processing 1*. 1994.