Semantic Parsing for Vietnamese: A Cross-Lingual Approach

Tessa Pham

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Arts in Computer Science and Linguistics.

Advisor: Jane Chandlee

April 17, 2020

Abstract

As the 16th most spoken language in the world, Vietnamese is surprisingly lacking in annotated resources for natural language processing (NLP) and computational linguistics. Due to this shortcoming, research on NLP tasks that involve the language has been limited, including semantic parsing (i.e., the task of converting a natural language utterance to a logical form). This thesis explores a cross-lingual approach to semantic parsing, i.e., adapting frameworks available in a high-resource language like English to represent the meanings of Vietnamese sentences, and applies it to build an experimental parser that can auto-generate semantic annotations for Vietnamese. First, we studied the structures of three meaning representation (MR) schemes that are among the most cross-linguistically robust, all of which were designed in English. The performance of each MR was evaluated based on how well it could be applied to annotate a Vietnamese sentence, which indicates its cross-lingual potential, and then discussed in relation to the others. After examining available resources for the three MR schemes, we chose Universal Conceptual Cognitive Annotation (UCCA) to use for our semantic parser that is essentially a machine learning (ML) model. A process to convert UCCA parses of English sentences to parallel Vietnamese sentences was designed by leveraging bitext word alignment, which enabled us to create a dataset containing semantic parses of over 38K Vietnamese sentences for training and testing. The results produced by our final model also revealed insightful areas of improvement for the cross-lingual semantic parsing task.

Table of Contents

1. Introduction

This section explains the motivation behind my thesis topic and introduces the task of *semantic parsing*.

1.1. Motivation

When linguistic or NLP research is done on Vietnamese, the first step typically involves constructing an annotated corpus since very few gold standards exist for the language. Some projects focused exclusively on building either a syntactically or semantically annotated corpus in hopes that it could be used in future work, but the quality of such corpora is difficult to measure due to existing contentions in the field over basic standards for the language. At the moment, given its lack of resources, Vietnamese is not too accessible for even native linguists, not to mention foreign academics who may be interested in the language. This limitation hinders it from being widely studied, and, without many research results, Vietnamese is not being adapted to modern technologies as quickly as one would hope. More specifically, research has been limited on certain NLP tasks for the language, including semantic parsing. To facilitate such tasks without in-language resources, natural language researchers have experimented with cross-lingual approaches that rely on frameworks provided in a high-resource language like English. For semantic parsing in particular, the cross-lingual approach first aligns a parallel text (also called a bitext) given in a low-resource language (called the source language) and a higher-resource one (called the target language), then parses semantics for the half in the target language (in which MRs are readily available), and finally matches the annotations back to the half in the source language. The first half of this thesis (chapters 1-4) gives background on three frameworks that have been claimed to be cross-linguistically robust, investigates how each of them can be applied to annotate Vietnamese semantics, and analyzes how well they handle certain differences between English and Vietnamese that may pose difficulties for cross-lingual semantic parsing. The latter half (chapter 5-8) explains our methodology to implement a semantic parser for Vietnamese using this cross-lingual approach, which includes

discussion on available data and resources, the use of bitext word alignment, generating data for training, and finally fitting and evaluating the model. As of this writing, no semantic parser for the language has been developed yet. Results from this study will inform future work on adapting a cross-lingual approach to other NLP tasks and hopefully provide another useful way to auto-generate annotated resources for not only Vietnamese but also other low-resource languages.

1.2. Semantic Parsing

Semantic parsing is an NLP task of representing the meaning of a natural language utterance in a machine-understandable form, known as a *logical form*. Essentially, the task involves extracting the literal meaning of an utterance, which technically is creating and assigning semantic representations to its constituents (Jurafsky and Martin, 2009). Semantic parsing can be applied to machine translation (i.e., software that translates text or speech from one language to another) and question answering (i.e., systems that automatically answer a question posed in human natural language), among many other disciplines.

There are two types of semantic parsing: shallow and deep. *Shallow semantic parsing*, also known as *semantic role labeling (SRL)*, involves identifying and labeling entities in a sentence with categorical roles. The theoretical basis for SRL comes from *frame semantics*, whose idea is that a word activates a *frame* of related concepts and roles. The concept of frames will be re-encountered later on in one of the MRs we are going to discuss. *Deep semantic parsing*, also known as *compositional semantic parsing*, is the task of producing meaning representations for expressions that are significantly more complex and contain possible ambiguities. For example, shallow semantic parsers can parse sentences like "show me flights from Philadelphia to San Diego" by classifying the intent as "list flights" and supplying answers to "source" and "destination" with "Philadelphia" and "San Diego" respectively. However, shallow semantic parsing cannot parse arbitrary compositional sentences, like "show me flights from Philadelphia to anywhere that has connecting flights to Hanoi". Deep semantic parsing handles such sentences using a formal meaning representation

scheme/language, to be presented in the next chapter. With the aim to provide thorough semantic annotations, this thesis is concerned with deep semantic parsing.

The most challenging problem that hinders straightforward parsing of a sentence's logical form is ambiguity (Tucker, 2002). As we already know, some words have many senses, and some can take on different senses in different parts of speech. To handle this problem, linguists use an *ontology*, a classification scheme for objects, as an attempt to categorize and represent word meanings. According to Tucker in his NLP notes (2002), this scheme was first proposed by Aristotle, and many of whose original classifications are still being used today: action, position, event, state, substance, affection, quantity, quality, relation, place, ideas, concepts, plan, situation, time, and so on. To further indicate the relations between objects and actions, *case grammars* are used to analyze the structure of sentences with *case roles* (also known as *semantic/thematic roles*) of nouns in relation to predicates. These roles typically include, but are not limited to, *agent*, *theme*, and *instrument*, which are often employed in representing the semantics of an utterance.

Most semantic parsing research for Vietnamese is restricted to question answering only, which does not capture any large part of a language. To expand on previous efforts, this paper studies deep semantic parsing, which can serve as the backbone for a comprehensive semantic parser for the language.

2. Meaning Representations (MRs) and Their Computational Constructions

This section provides a comprehensive background on *meaning representation (MR)*, an approach to semantics this paper will be centered on, and explains the structures and characteristics of the three MRs to be applied to our cross-lingual semantic parsing task.

2.1. Definition

Theoretically speaking, developing a full semantic framework for a natural language would require a complete and uniform theory of how people think and communicate ideas, but this is virtually unattainable since the process of verbalizing one's thoughts

differs from person to person, let alone across languages. Consequently, natural language researchers have decided to capture semantics in formal structures called *meaning representations* (MRs), which are more modest but pragmatic approaches (Tucker, 2002). The frameworks used to define the syntax and semantics of these MRs are called *meaning representation languages/schemes*, which are also referred to as MRs for short. In this paper, MR will be used interchangeably to mean either a representation language or the semantic representation of a specific sentence. By formally representing semantics, MRs also bridge the gap between the *raw linguistic inputs* and the *non-linguistic knowledge of the world* needed to perform tasks that involve the meanings of these inputs (Jurafsky and Martin, 2009). To demonstrate this point, let us consider the task of answering the question "Which country is Hanoi in?". The phonological, morphological, and syntactic representations directly accessible from the raw inputs (i.e., the question), are not enough to help us accomplish this task. We need other non-linguistic knowledge and pieces of information necessary to answer this question, which can be inferred from the meaning of the linguistic inputs: i.e., *the question asks for* a **country C** *given that* **Hanoi** (a **city**) is **in C**. Then, we know to search in a *list of countries* for a country whose *list of cities* contains Hanoi.

Historically, the most basic way to represent meaning is first-order logic (also known as first-order predicate calculus or FOPC) (Jurafsky and Martin, 2009), which covers predicate and quantification by using functions and quantified variables for non-logical objects (e.g., $\exists x$ to mean there exists an object represented by variable x). The meaning of a sentence encoded in FOPC typically has a predicate-argument structure. For example, we can translate the sentence "Vietnam is the country which Hanoi is in" into FOPC as follows:

(1) $\exists x, y: Country(x) \wedge Vietnam(x) \wedge Hanoi\ (y) \wedge IsIn(y, x)$

Or more simply without using variables:

(2) $Country(Vietnam) \wedge IsIn(Hanoi, Vietnam)$

In both (1) and (2), we can see the predicate-argument structure embedded in the FOPC representation: there are two predicates with function-like representations

$Country()$ and $IsIn()$, which correspond to "is the country" and "is in" in the original sentence; and placed within the parentheses of each predicate are its arguments.

To implement semantic parsing with a computational model, we need more advanced approaches to representing semantics. First and foremost, instead of the traditional logical form with lambda calculus, we want to use some modern MR that is designed with a goal to be computer-friendly, i.e., having a machine-readable representation (such as graphs, as we will see in the MRs presented later on). In addition, some modern MRs also experiment with treating the semantic representation of a sentence independently from its syntactic structure, which is often referred to as being *syntax-independent*. This concept may sound strange and unintuitive to linguists since semantics almost always goes hand-in-hand with syntax and, with that, syntax lends much support to disambiguating the meanings of complex sentences. Without relying on syntactic structures, semantic parsing models may run into certain difficulties and not gain as much success as if syntax were taken into consideration. However, the goal of syntax-independence is two-fold. First, it aims to assign the same representation to utterances with the same underlying meaning regardless of their possibly different syntactic forms. For example, the MRs for these sentences should be similar: "John went out with Mary yesterday", "Yesterday was when John went out with Mary", and "It was Mary who John went out with". Secondly, abstracting away from syntax (i.e., obscuring or ignoring syntactic forms and information) in the parsing process can significantly simplify a computational model and increase its portability across languages (since syntax is language-specific, accounting for it will make the model less flexible cross-linguistically). In contrast to syntax-independent models, the typical syntax-driven semantic parsing process is illustrated in Figure 1:

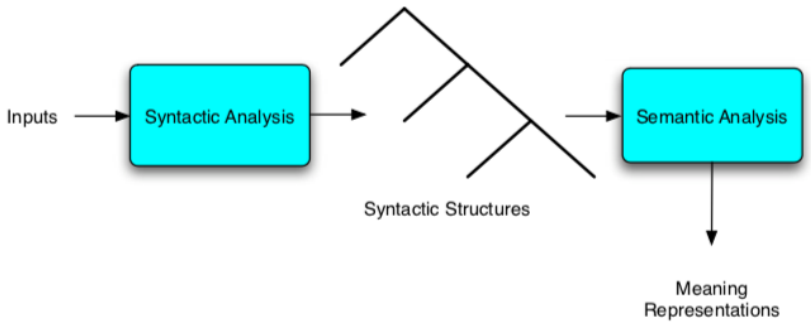

Figure 1. Syntax-driven semantic analysis (Jurafsky and Martin, 2009)

The following parts of this section introduce three different MRs: Universal Decompositional Semantics (UDS), Abstract Meaning Representation (AMR), and Universal Conceptual Cognitive Annotation (UCCA). While AMR and UCCA aim for syntax-independence, UDS hinges on syntactic dependencies. Each of them has its own characteristics, which will also be mentioned and discussed.

2.2. Universal Decompositional Semantics (UDS)

The first MR we are going to delve into is UDS, which is a set of semantic annotation protocols that are aimed for cross-linguistic robustness (White et al., 2016). It was developed by the Decompositional Semantics Initiative (Decomp), which incorporates *semantic decomposition* into its protocols, i.e., an idea that semantic annotation should be an aggregation of many simple questions about words or phrases in context that are easy for naïve native speakers to answer, thus allowing for crowd-sourced annotations and coverage of non-prototypical sentences (White et al., 2016).

2.2.1. Universal Dependencies (UD)

UDS was initially designed to augment the Universal Dependencies (UD) datasets with semantic annotations (White et al., 2016). These datasets are results of the UD project, which constructed a standard for syntactic dependency annotation that can be used across languages and provided more than 100 treebanks of over 70 languages (including Vietnamese) annotated using this standard (De Marneffe et al., 2014; UDS website, 2020). Figure 2 shows example UD syntactic parses for the sentence "The dog was chased by the cat" in four different languages.

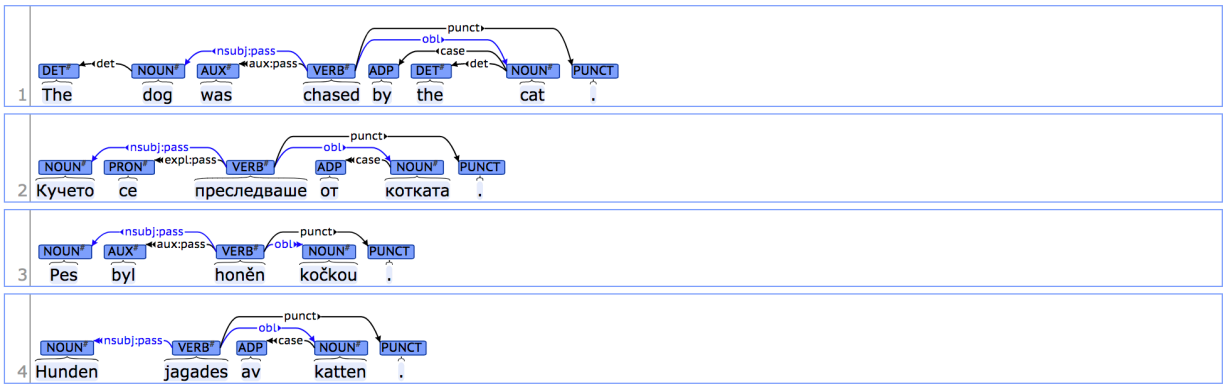

Figure 2. UD parses in four different languages[1]: (from top to bottom) English, Bulgarian, Czech, and Swedish

The dark blue box over each token contains the POS tag for that token. An arrowed link that goes from one token to another indicates the dependency relation between the two tokens. For example, in the first sentence in Figure 2, the link from "dog" to "the" is labeled *det*, indicating that "the" is the determiner of "dog". Relations are often written as $relation(dependent, governor)$, so the relation in this example can be expressed as $det(the, dog)$. We will re-encounter this annotation in the demonstration of the UDS parsing process. The blue links represent the main relations, which are *nsubj* and *obl* in all four parses. In sentence 1, $nsubj{:}pass(dog, chased)$ means that "dog" is the subject of the predicate "chased" and that the predicate is in passive voice (*pass* stands for passive), and $obl(cat, chased)$ means that "cat" is an oblique argument of "chased", i.e., functioning as an adverbial attached to the verb "chased". Explanations for other relations marked on the black edges can be found in UD's online documentation[2] of its relations.

UD parses are in fact dependency trees, i.e., rooted directed graphs in which edges explicitly represent syntactic dependency relations. The parses in Figure 2 are the linearized format of such trees. Figure 3 shows what the dependency tree for the sentence "The dog was chased by the cat" looks like. The number beside each token is the order of that token in the sentence. UD parses always assign the main predicate

[1] https://universaldependencies.org/introduction.html
[2] https://universaldependencies.org/u/dep/index.html

the relation $root(None, main\ predicate)$, which explains the edge $root(None, chased)$ in the example tree.

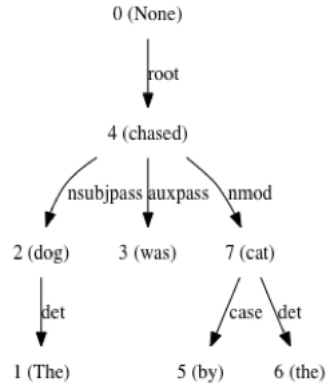

Figure 3. Dependency tree for "The dog was chased by the cat"
(generated using *StanfordDependencyParser* and *graphviz* packages in Python)

2.2.2. PredPatt

In order to understand the structures of UDS, we first need to explore PredPatt, an MR that acts as a scaffold of UDS and is mostly used for shallow semantic analysis. It is used as a pre-processing tool to identify the predicate-argument structures from UD parses. Output from PredPatt is fed as input to the annotation protocols of UDS (White et al., 2016).

The PredPatt process takes in a UD parse (like one in Figure 2), identifies predicate and argument roots, resolves arguments, extracts predicates and argument phrases, then does optional post-processing (White et al., 2016). Predicate and argument roots are nodes in the dependency tree (i.e., a non-linear representation of the UD parse, as shown in Figure 3) and are identified by edges in the parse. For example, $nsubj(s, v)$ and $obj(o, v)$ (i.e., o is the direct object of v) indicate that v is a predicate root, and s and o are argument roots (White et al., 2016). After the identification step, PredPatt handles missing arguments due to syntactic structures by applying argument resolution rules. An example case that calls for argument resolution is when predicates appear in a conjunction: e.g., "Betty cooked and relaxed" does not have an edge connecting "relaxed" to its subject "Betty". Moving on to

predicate extraction, PredPatt uses text order to derive names for complex predicates. For example, let us consider the sentence "[I] did not **meet** [John] [at Starbucks] yesterday", the predicate of which is initially identified to be "meet". PredPatt lifts mark and case tokens (e.g., "at") out of the argument subtree, then adds any adverbial modifiers (e.g., "yesterday"), auxiliaries (e.g., "did"), and negation (e.g., "not"), producing a 3-slot predicate named (?a **did not meet** ?b **at** ?c **yesterday**). To extract argument phrases, PredPatt filters tokens from the dependency subtree to simplify it, e.g., by removing relative clauses. Its default set of filters makes sure meaning is preserved as certain modifiers are omitted. We will see the whole PredPatt process in action when UDS is applied in §3.

PredPatt offers three synonymous ways to represent meaning: flat, graph, and linearized. Figure 4 shows the sentence "30 people were reported dead in one block of flats which was hit by a storm surge" represented in all three formats.

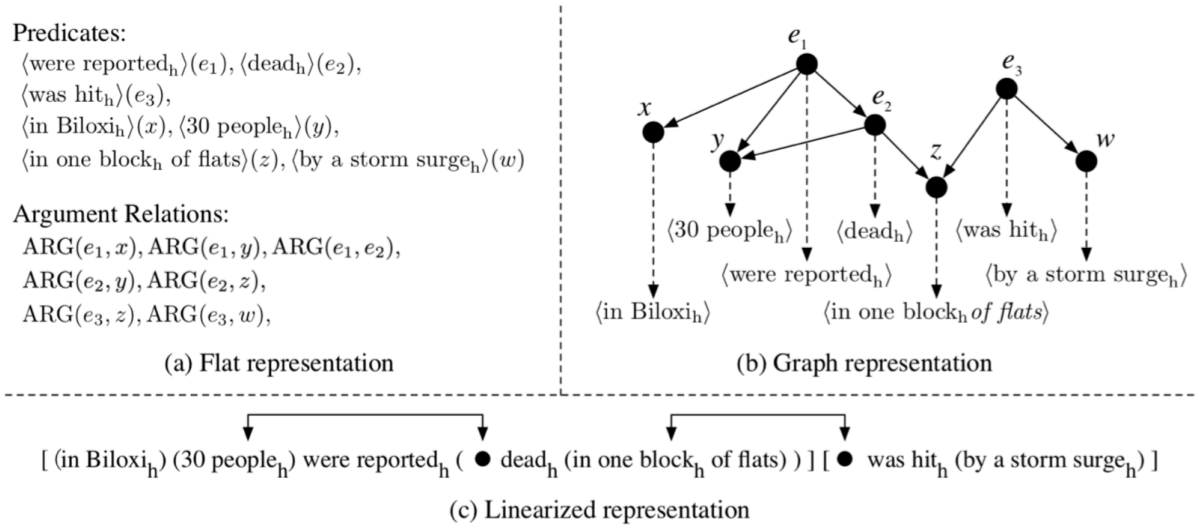

Figure 4. 3 formats to represent semantics in PredPatt
(Zhang et al., 2018)

The flat representation contains a bag of predicates and a bag of argument relations, each of which pairs a predicate root with one of its dependent arguments. The graph notation is simply a dependency tree, in which predicate-argument relations are modeled with directed edges. To obtain the linearized representation, starting from the root node of the dependency tree (i.e, "reported$_h$"), we perform an in-order

traversal (left – root – right) on its spanning tree. We place square brackets to mark boundaries of a predicate span and parentheses for boundaries of an argument span. The syntactic head of each span is denoted by the subscript "h". When coreference occurs within a sentence (i.e., when a predicate or an argument refers to a preceding node), the symbol "•" is inserted at the beginning of that span and a link is added between "•" and the preceding node to which it refers.

2.2.3. Semantic Proto-Roles (SPR)

One last concept to grasp before we discuss UDS is semantic proto-roles (SPR), created by Dowty in 1991. Dowty came up with SPR to handle the problems encountered in delineating boundaries between theta-roles, one of which he termed *role fragmentation*, i.e., the situation where attempts to create fine-grained thematic roles lead to an unreasonably large set of specialized roles that cannot be generalized (Dowty, 1991; Reisinger et al., 2015). With SPR, instead of being assigned categorical roles, semantic arguments are scored over fine-grained properties that constitute a role, i.e., features such as whether the entity in question changes state or causes an event to happen (Reisinger et al., 2015). In other words, semantic annotations are *property-based*. Dowty divides role properties into two groups: Proto-Agent properties and Proto-Patient properties (refer to table in Figure 5). Expectedly, an argument that has more Proto-Agent properties is considered more agent-like, while one with more Proto-Patient properties is more patient-like. It is possible for an argument to have some properties of each group or even none of them (Reisinger et al., 2015). We will see more concretely what this looks like when we get to the UDS representations.

Proto-Agent properties	Proto-Patient properties
a. volitional involvement	f. changes state
b. sentience (/perception)	g. incremental theme
c. causes change of state	h. causally affected
d. movement (relative)	i. stationary (relative)
e. independent existence	j. no indep. existence

Figure 5. Dowty's two groups of proto-role properties (Reisinger et al., 2015)

Just like we have semantic role labeling (SRL) for theta-roles, with SPR comes a new task: semantic proto-role labeling (SPRL), which replaces coarse-grained categorical roles with real-valued scores of fine-grained properties inspired by Dowty's approach, a set of which can be thought of as a vector. This task is one of the main steps in UDS. With that said, UDS completely discards categorical roles and gears towards a vector-space model. For gold-standard SPR-annotated corpora, SPRL is done manually by trusted annotators from Amazon Mechanical Turk answering a set of questions such as in Figure 6. (These questions can vary based on which properties the model designers deemed important in defining their semantic arguments.)

Role property	How likely or unlikely is it that...
instigation	ARG caused the PRED to happen?
volition	ARG chose to be involved in the PRED?
awareness	ARG was/were aware of being involved in the PRED?
sentient	ARG was/were sentient?
change of location	ARG changed location during the PRED?
-exists as physical	ARG existed as a physical object?
existed before	ARG existed before the PRED began?
existed during	ARG existed during the PRED?
existed after	ARG existed after the PRED stopped?
change of possession	ARG changed possession during the PRED?
change of state	ARG was/were altered or somehow changed during or by the end of the PRED?
-stationary	ARG was/were stationary during the PRED?
-location of event	ARG described the location of the PRED?
-physical contact	ARG made physical contact with someone or something else involved in the PRED?
was used	ARG was/were used in carrying out the PRED?
-pred changed arg	The PRED caused a change in ARG?
+was for benefit	PRED happened for the benefit of ARG?
+partitive	*Only* a part or portion of ARG was involved in the PRED?
+change of state continuous	The change in ARG happened throughout the PRED?

Figure 6. An example set of property questions for SPRL (White et al., 2016)

2.2.4. UDS Structures and Components

Just like PredPatt, UDS provides three interchangeable representations for different purposes: flat, graph, and linearized, which are created for different purposes and are interconvertible (Zhang et al., 2018). Demonstration of the flat format will be omitted

since it is not used as much as the other two in computational tasks. Figure 7 shows the sentence "30 people were reported dead in one block of flats which was hit by a storm surge" represented in the graph format.

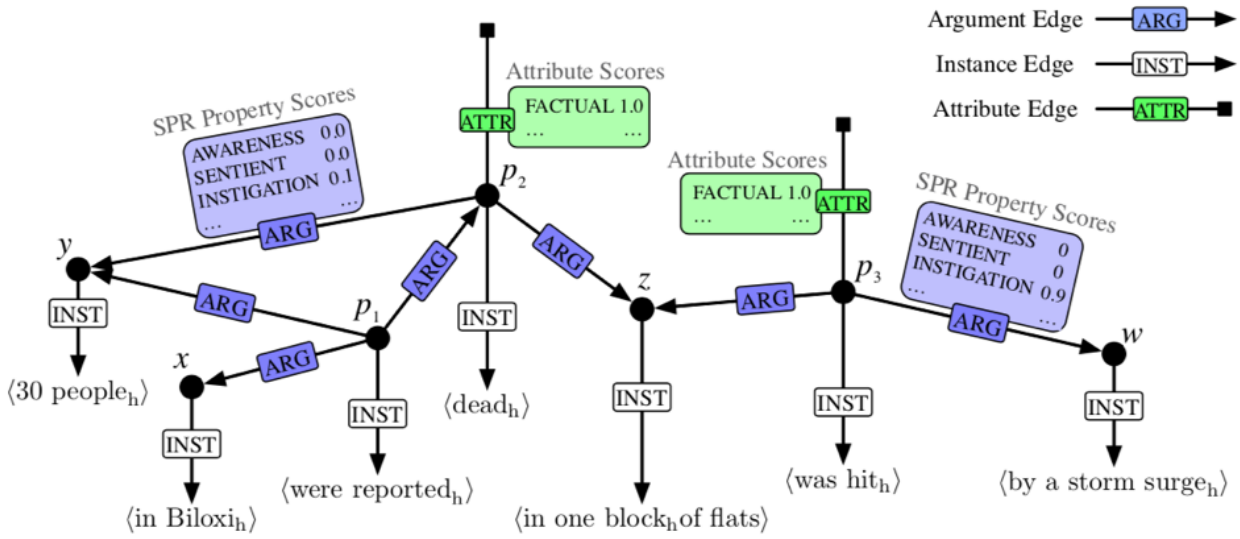

Figure 7. UDS graph representation (Zhang et al., 2018)

As we can see, the skeleton of this graph (i.e., nodes, edges, and variables) is indeed PredPatt's graph representation for the sentence in Figure 4(b). Several undirected edges are added to represent attributes of variables such as event factuality and word senses (Zhang et al., 2018), which we will not go into. For distinction among edges, each is explicitly marked with a label to indicate its type. According to Zhang and colleagues (2018), there are three types of edges. The first one is argument edges which model relations between predicate-argument pairs. For deeper semantic analysis, SPR properties (as explained in 2.2.3) can be attached to argument edges. The second type is instance edges which link variables to their instances in the original language. Finally, attribute edges further describe variables as aforementioned. Due to the complexity of extracting these attributes for variables, attribute edges are intentionally ignored when we apply UDS to manually annotate our example in a later section.

Despite quite some additions to the graph representation of UDS, its linearized representation remains almost exactly the same as PredPatt's. The only minor difference is that coreference links are explicitly labeled.

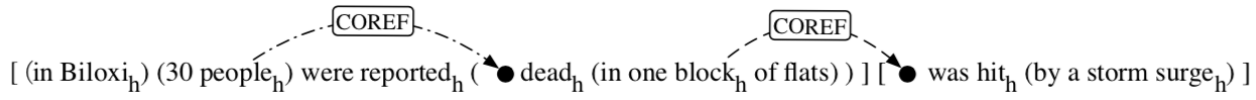

Figure 8. UDS linearized representation (Zhang et al., 2018)

In summary, the UDS MR scheme essentially builds upon UD parses, using PredPatt to extract predicate-argument relations and SPR for deeper semantic analysis of each relation. The diagram in Figure 9 illustrates how UDS enhances a UD parse with decompositional semantic annotations, which was its initial goal as mentioned at the beginning of this chapter.

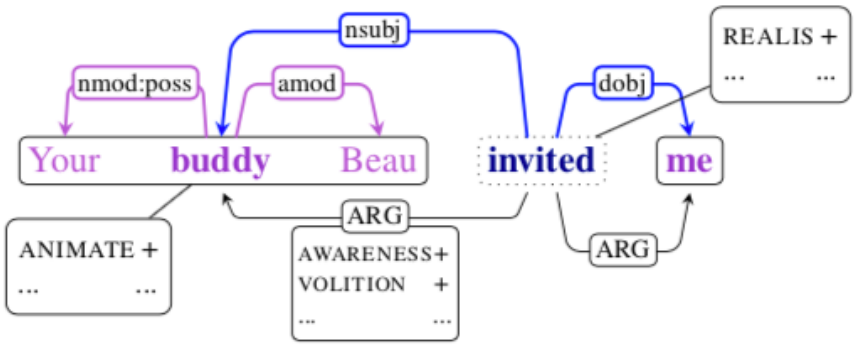

Figure 9. Decompositional semantic annotation layered atop UD syntax (White et al., 2016)

2.2.5. UDS Characteristics

As we have seen from the construction of UDS, its design aims to provide as close an analysis to the syntax of natural language as possible, which is opposite to the mutual goal of AMR and UCCA to abstract away from syntactic idiosyncrasies, to be introduced in the next subsection.

2.3. Abstract Meaning Representation (AMR)

The second MR we are going to look at is AMR. AMR is a semantic representation language initially designed for sembanking, i.e., constructing a semantic treebank (sembank) of whole-sentence semantic structures. The creators of this sembank wanted to simplify the task of semantic annotation: moving from separate annotations for named entities, semantic relations, etc. to whole-sentence annotations, and thus created AMR specifically to handle such a task (Banarescu et al., 2013). From here on, the term AMR will be used interchangeably to mean either the representation language itself or the meaning of a specific sentence represented in this language. AMR aims to abstract away from syntactic forms so that sentences with the same underlying meaning get assigned the same AMR, the idea of which is elaborated in 2.3.1. Moreover, it does not provide any rules for deriving meanings from utterances and vice versa, which not only simplifies sembanking but also allows room for researchers to freely explore and interpret the relations between utterances and meanings (Banarescu et al., 2013).

2.3.1. Frameset

Due to its frequent use in AMR, it is important that we understand what a frameset is before diving deeper into AMR structures. A frameset defines the argument roles of a verb, which are denoted as *arg0*, *arg1*, … depending on how many arguments the verb takes, and is denoted by the verb itself (e.g., "want-01"). A verb may have multiple word senses, which are numbered arbitrarily within a list of framesets (such a list should be provided in any model or tool that uses framesets). The number after the dash in each frameset name therefore indicates which word sense of the verb is at play. Figure 11 gives an example of how a full frameset is defined.

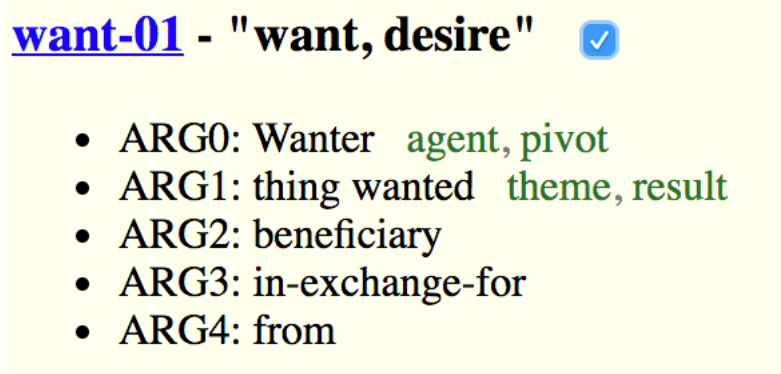

Figure 10. Frameset for "want-01" provided by the AMR Editor[3]

Later on, we will see that framesets are employed even in phrases that contain no verbs. For example, the phrase "bond investor" can be represented with the frame "invest-01" (Banarescu et al., 2013). This is due to the fact that AMR intends for framesets to represent not verbs but *concepts*, which can be realized as either verbs, nouns, or adjectives. For example, the concept "describe-01" can stand for a verb ("describe") or a noun ("description"). As we will see in the next part, this serves AMR's goal of abstracting away from syntax nuances that do not affect the basic meaning of the sentence, i.e., being syntax-independent.

2.3.2. Syntax Independence

The frameset "describe-01" has three predefined arguments: *arg0* is the describer, *arg1* is the object being described, and *arg2* is what it is described as. By regarding "describe-01" as a concept, as mentioned above, we can represent both "describe" (verb) and "description" (noun) with this frameset, so that if a sentence "*arg0* describes *arg1* as *arg2*" is paraphrased as "*arg0*'s description of *arg1* is *arg2*" for example, both will still get the same AMR. In fact, Figure 11 shows an example of AMR being able to represent three syntactically different sentences that have the same underlying meaning.

[3] https://www.isi.edu/cgi-bin/div3/mt/amr-editor/load-amr-v1.7.cgi

```
(d / describe-01
  :arg0 (m / man)
  :arg1 (m2 / mission)
  :arg2 (d / disaster))

The man described the mission as a disaster.
The man's description of the mission:
  disaster.
As the man described it, the mission was a
  disaster.
```

Figure 11. Three syntactically-diferent sentences getting the same MR (Banarescu et al., 2013)

As we can see, AMR chooses to ignore syntactic information such as whether the concept of describing something is realized as a verb ("describe") or a noun ("description") or whether the verb "describe" acts as the main predicate or appears in a subordinate clause (like in the third sentence), in order to focus on representing the main idea being conveyed. Abstracting away from such syntactic nuances still ensures a complete semantic representation as long as we can identify the argument roles of our main concept, which in this case are "man" for *arg0*, "mission" for *arg1*, and "disaster" for *arg2*. (Note that determiners are also glossed over.) This is exactly how AMR is designed to aim for syntax independence.

2.3.3. AMR Structures and Components

AMR structures are rooted, directed, edge-labeled, and leaf-labeled graphs (Banarescu et al., 2013). In order for its representations to be both computer- and human-friendly, AMR provides two notations: graph for computer processing and PENMAN for human reading and writing. Figure 12 shows the sentence "The boy wants to go" annotated in these formats and in first-order logic for comparison.

LOGIC format:

∃ w, b, g:
instance(w, want-01) ∧ instance(g, go-01) ∧
instance(b, boy) ∧ arg0(w, b) ∧
arg1(w, g) ∧ arg0(g, b)

AMR format (based on PENMAN):

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
          :arg0 b))
```

GRAPH format:

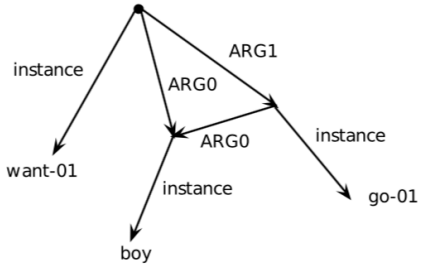

Figure 12. Equivalent representations for the meaning of "The boy wants to go" (Banarescu et al., 2013)

The logic format utilizes FOPC annotation but looks quite different: functions are only of two types, $instance(variable, frameset)$ and $arg(variable1, variable2)$. The first function type assigns a *variable* to either a word (e.g., "boy") or a frameset (e.g., "want-01"). The second function type defines a predicate-argument relation: e.g., *arg0(w, b)* means that *b* (i.e., "boy") is *arg0* (i.e., the wanter according to Figure 10) of the predicate *w* (i.e., "want-01"). AMR uses variables for entities, events, properties, and states, as we can see with *w, b, g* standing for "want-01", "boy", "go-01" respectively in the example above. These variables are replaced by nodes in the graph notation.

Following the explanation of the logic format, the PENMAN notation is now pretty straightforward. Note that the arguments are specified on indented lines below the predicate they belong to. Interestingly, the last line (i.e., ":arg0 b") indicates that "boy" is also an argument of the predicate "go". This is a case of an entity having

multiple roles in a sentence, which the PENMAN notation handles with variable reuse (Banarescu et al., 2013).

AMR's graph notation replaces functions in the logic format with edges and variables with nodes. Same as PENMAN, this notation needs to depict the entity "boy" being an argument of both "want" and "go", for which it adopts reentrancy, i.e., a node having multiple parents. As we can see in Figure 12, the node with two incoming ARG0 edges in fact is the replacement of the variable *b* for "boy".

In AMR, entities are often called *concepts* and concepts are linked together by *relations*. AMR concepts are either English words (e.g., "boy"), framesets (e.g., "want-01"), or special keywords. These keywords comprise of entity types (e.g., "date-entity" or "world-region"), quantities (e.g., "monetary-quantity" or "distance-quantity"), and logical conjunctions (e.g., "and"). AMR relations indeed include whatever we referred to as "functions" in the explanations above. In the graph notation, relations are marked on the edges and concepts are labeled on the leaves (i.e., terminal nodes). In the PENMAN notation, concepts are written such as "b / boy", which means an instance "b" of the concept "boy". Taking the following representation for example:

(3) (d / die-01
:location (p / park))

The relation *:location* connects two entities "d / die-01" and "p / park" to mean there was a death in the park. According to Banarescu and colleagues (2013), AMR has approximately 100 relations. This comprehensive set of concepts and relations allows AMR the ability to consistently represent all possible sentences (Banarescu et al., 2013).

2.3.4. AMR Characteristics

AMR is heavily biased towards English, given the list of framesets that are exclusively defined in English. As a result, generating AMRs for languages other than English requires a good amount of pre-processing: named entity recognition, semantic role labeling, word sense disambiguation, among other tasks depending on the intended purpose for the AMR-annotated corpus (Ha and Nguyen, 2019). This

inflexibility is evidenced in the work of Ha and Nguyen (2019), "A Case Study on Meaning Representation for Vietnamese", in which the researchers adapted AMR to semantically annotate *The Little Prince* corpus in Vietnamese and had to design additional components for their model, despite AMR's rich set of over 100 relations.

As noted by Zhang and colleagues (2018) when comparing AMR with UDS, AMR embraces underspecification by choosing not to annotate certain aspects of meaning (e.g., no quantifier scope due to dropping of determiners). While underspecification allows for tractable computational models, it can pose limitations with the lack of representation for quantifier scope, co-references, tense, aspect, and quotation marks (Ha and Nguyen, 2019).

2.4. Universal Conceptual Cognitive Annotation (UCCA)

UCCA is a semantic annotation scheme heavily influenced by *basic linguistic theory* and cognitive linguistic theories. Coined by R. M. W. Dixon (2006), basic linguistic theory is a term for the theoretical framework and basic concepts widely employed in language description and linguistic typology. UCCA aims to provide a representation that can be applied across languages by satisfying two conditions: it captures major semantic phenomena and is independent of syntactic idiosyncrasies (Sulem et al., 2015). Instead of building on other annotation layers like UDS, UCCA extracts semantic representations directly from the raw linguistic inputs (Abend and Rappoport, 2013), the process for which we will see in the next sections.

2.4.1. UCCA Structures and Components

UCCA regards a sentence as a sequence of Scenes containing relations and participants. The sentence is divided into *units*, each referring to a relation, a participant in a relation, or a relation along with its participants. (Definitions for Scene, participant, and different types of relations will be given in the next section.) The meanings of UCCA-annotated sentences are represented in directed acyclic graphs (DAGs) (i.e., graphs with directed edges and no cycles), whose nodes represent

these units and thus are uniformly referred to as units. Figure 13 shows an example UCCA graph for the sentence "John kicked his ball":

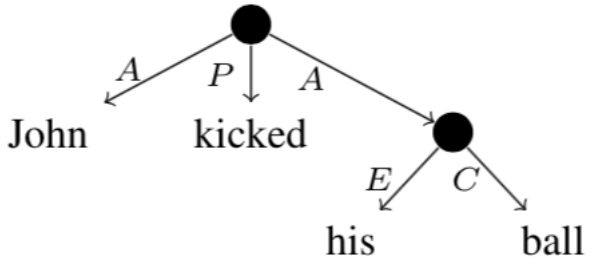

Figure 13. UCCA annotation graph for "John kicked his ball"
(Abend and Rappoport, 2013)

As we can see from the example, terminal nodes (i.e., leaves or childless nodes) are sentence tokens, which can be either words or multi-word units. Non-terminal nodes are semantic units that represent several elements cognitively considered as a single entity. Let us take as examples the two non-terminal units in Figure 13 (i.e., the black dots), and call the one on the first level (whose outgoing edges are denoted with the letters *A*, *P*, *A* from left to right) "unit J" and the one on the second level (whose outgoing edges are denoted with the letters *E* and *C*) "unit K". Unit K represents "his" and "ball" together, which are viewed as one entity according to some semantic or cognitive consideration (Abend and Rappoport, 2013). The same line of reasoning also applies to unit J, which actually represents the whole sentence.

Edges are labeled with categories (e.g., *A*, *P*, *E*, *C* in Figure 13) which represent the child unit's role in forming the semantics of the parent unit. For example, the terminal unit "his" plays the role *E* (for Elaborator) and the terminal unit "ball" plays the role *C* (for Center) in forming the meaning of unit K, i.e., "his ball". Moreover, unit K representing "his ball" plays the role *A* (for Participant) in forming the meaning of unit J, i.e., the whole sentence. We will explore what each of these categories means in the next section. From the aforementioned example, we can see that the complete structure of a unit is represented by its outgoing edges and their categories, while its incoming edges represent the roles the unit plays in the relations it participates in (i.e., the relations represented by its parent nodes) (Abend

and Rappoport, 2013). In fact, a unit may have more than one parent node since it is allowed to participate in more than one relation, which happens when the unit plays a role (i.e., is referred to) in multiple Scenes.

UCCA annotation graph can also be linearized. The graph in Figure 13 has a linearized representation as follows:

(4) John$_A$ kicked$_P$ [his$_E$ ball$_C$]$_A$.

Each token is subscripted with the role it plays in a superordinate relation (i.e., a parent node in the UCCA graph). A multi-word unit that participates in some relation as a single entity is marked with square brackets. For example, the whole entity "his ball" is a Participant (*A*) in the Process (*P*) "kicked" even though it contains two words, so brackets are placed around this multi-word. Additionally, the terminal units "his" and "ball" play different roles (i.e., Elaborator (*E*) and Center (*C*) respectively) within "his ball" (i.e., their superordinate relation) and are thus further annotated within the square brackets.

UCCA is constructed as a multi-layered framework where each layer specifies the relations it encodes, which determines which units will be formed. This feature allows for open-ended extensions for deeper semantic analysis (Abend and Rappoport, 2015). Most research has been focused on the foundational layer, which covers predicate-argument structures and their interrelations and thus is guaranteed to be able to parse all grammatical sentences. At the moment, it is the only layer in which corpora are annotated, providing the most resources for analysis (Hershcovich et al., 2019).

2.4.2. The Foundational Layer

The foundational layer is designed to cover every possible sentence, such that each word in the sentence participates in at least one unit (Abend and Rappoport, 2013). Table 1 summarizes all the categories in the foundational layer of UCCA:

		Scene Elements
P	**Process**	The main relation of a Scene that evolves in time (usually an action or movement).
S	**State**	The main relation of a Scene that does not evolve in time.
A	**Participant**	Scene participant (including locations, abstract entities and Scenes serving as arguments).
D	**Adverbial**	A secondary relation in a Scene.
		Elements of Non-Scene Units
C	**Center**	Necessary for the conceptualization of the parent unit.
E	**Elaborator**	A non-Scene relation applying to a single Center.
N	**Connector**	A non-Scene relation applying to two or more Centers, highlighting a common feature.
R	**Relator**	All other types of non-Scene relations: (1) Rs that relate a C to some super-ordinate relation, and (2) Rs that relate two Cs pertaining to different aspects of the parent unit.
		Inter-Scene Relations
H	**Parallel Scene**	A Scene linked to other Scenes by regular linkage (e.g., temporal, logical, purposive).
L	**Linker**	A relation between two or more Hs (e.g., "when", "if", "in order to").
G	**Ground**	A relation between the speech event and the uttered Scene (e.g., "surprisingly").
		Other
F	**Function**	Does not introduce a relation or participant. Required by some structural pattern.

Table 1. Complete set of categories in UCCA's foundational layer

(Hershcovich et al., 2019)

As mentioned in the previous section, UCCA views a sentence as a collection of Scenes, each of which describes either a movement, an action, or a state that is persistent in time. Structurally, every Scene contains one main relation, which is either a Process (P) or a State (S), and one or more Participants (A). Figure 14 displays a UCCA annotation graph for an example sentence, "After graduation, John moved to Paris".

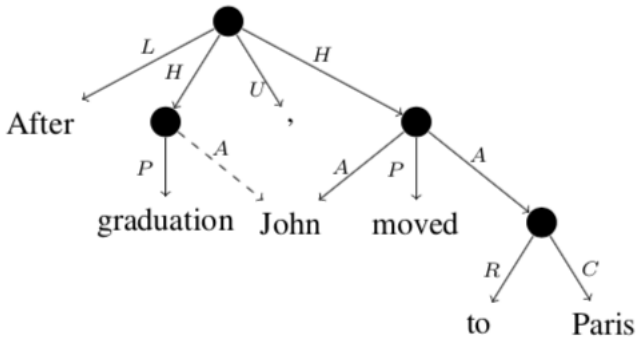

Figure 14. An example sentence represented in a UCCA graph

(Hershcovich et al., 2019)

The sentence shown in Figure 14 contains two Scenes whose main relations are two Processes: "graduation" and "moved". We can see more clearly why "graduation" is also counted as a Process if we paraphrase the sentence as "After he graduated, John Moved to Paris". The first Scene has "John" as a Participant while the second has

both “John” and “to Paris” as Participants. As we can see in the graph above, inter-Scene relations and multi-word expressions are described by further categories, such as Parallel Scene (H) (“John moved” happens after “graduation”, i.e., the two Scenes are linked temporally to each other), Relator (R) (for “to” in the multi-word expression “move to Paris”), and Center (C) (for “Paris” in the same expression). These categories additional to the main elements (i.e., Process (*P*), State (*S*), Participant (*A*)) account for the structures of more complex arguments (Hershcovich et al., 2019). The category *U* is for punctuations. Note that there is a dotted edge marked with the category *A* going into the terminal unit “John”, which indicates that “John” is an *implicit* Participant (*A*): “graduation” without a possessive modifier necessarily implies that it is an event related to the subject, i.e., “John”, which in turn means that “John” must play a role in “graduation”.

2.4.2. Characteristics

UCCA has been proven to be portable, i.e., able to be applied to different languages, and stable, i.e., able to preserve structure after translations (Sulem et al., 2015), and is therefore a highly potential candidate for cross-lingual semantic parsing. These claims are made after Sulem and colleagues tried applying UCCA to French and realized that the MR scheme did well in capturing the phenomena specific to French grammar, such as reflexive verbs (2015). In the next chapter, we will see how UCCA performs in handling several phenomena in Vietnamese syntax that may pose difficulties for semantic parsing.

3. Cross-Lingual Semantic Annotation for Vietnamese

After discussing all three MRs, we finally explore how semantic annotation for Vietnamese is done cross-lingually. Note that we will do manual annotation using our knowledge and understanding of each MR instead of relying on any computational model, since the linguistic part of this thesis is not concerned with building a parser yet but how an MR can be applied to represent the meaning of a

Vietnamese sentence. Afterwards, we will evaluate the performance of each MR based on criteria introduced in 3.2.

3.1. Background

Vietnamese is an SVO and tonal language, containing 6 tones in total. The language is considered isolating, i.e., having a very low morpheme-per-word ratio. In fact, it has the lowest possible ratio, allowing only a single morpheme per word. This characteristic of the language indeed poses a lot of interesting problems for the semantic parsing task.

3.2. Syntax-Driven Semantic Analysis

In this section, we will attempt to do semantic parsing for Vietnamese using the traditional rule-based syntax-driven approach. Then, we will list out challenges encountered while using this approach. These challenges will then be discussed in the next section, with background given on how each phenomenon manifests in Vietnamese. Moving forward, they also will serve as criteria for evaluating how each MR performs when applied cross-lingually.

Let us start by analyzing an example sentence from *The Little Prince*:

(5) It was a picture of a boa constrictor in the act of swallowing an animal.

(6) Nó vẽ một con trăn đang nuốt một con thú.

(7) It draw one CLF boa constrictor AUX.PRS swallow one CLF animal.

Sentences (5) and (6) form a parallel text (i.e., bitext) between English and Vietnamese, and (7) is the Leipzig gloss of (6). In (7), constituents that are not translatable to English have their functions noted using standard abbreviations: CLF for classifier and PRS.AUX for present tense auxiliary.

To jog our memory, syntax-driven semantic analysis is often done given a set of syntactic rules, each of which corresponds to a semantic attachment. Table 2 is an example of how such a rule set is given.

Grammar Rule	Semantic Attachment
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NP \rightarrow Det\ Nominal$	$\{Det.sem(Nominal.sem)\}$
$NP \rightarrow ProperNoun$	$\{ProperNoun.sem\}$
$Nominal \rightarrow Noun$	$\{Noun.sem\}$
$VP \rightarrow Verb$	$\{Verb.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$
$Det \rightarrow every$	$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
$Det \rightarrow a$	$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$
$Noun \rightarrow restaurant$	$\{\lambda r.Restaurant(r)\}$
$ProperNoun \rightarrow Matthew$	$\{\lambda m.m(Matthew)\}$
$ProperNoun \rightarrow Franco$	$\{\lambda f.f(Franco)\}$
$ProperNoun \rightarrow Franco$	$\{\lambda f.f(Frasca)\}$
$Verb \rightarrow closed$	$\{\lambda x.\exists e Closing(e) \wedge Closed(e,x)\}$
$Verb \rightarrow opened$	$\{\lambda w.\lambda z.w(\lambda x.\exists e Opening(e) \wedge Opener(e,z) \wedge Opened(e,x))\}$

Table 2. Rule-based approach to semantic analysis for a limited vocabulary (Jurafsky and Martin, 2009)

Note that we will not use this table for our task since it involves a different vocabulary, but instead assume that we are already given a similar table with rules and vocabulary pertaining to our example above. Next, we create syntax trees for sentences (5) and (6), shown in Figure 15 and 16 respectively:

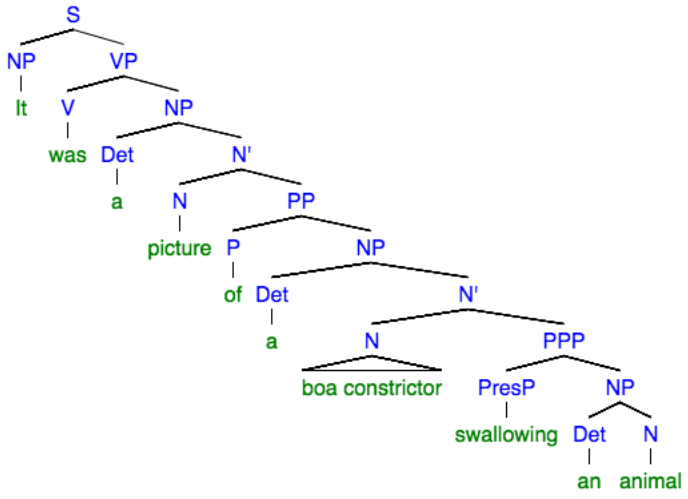

Figure 15. Syntax tree for sentence (5)

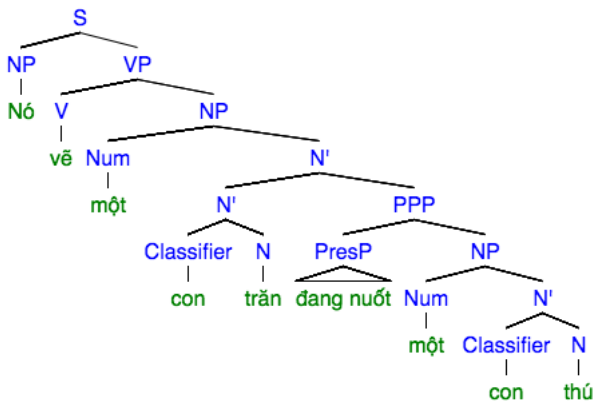

Figure 16. Syntax tree for the Vietnamese sentence (6)

From the syntax tree in Figure 15, we can generate FOPC semantic representation for sentence (5) as:

(8) $\exists x, y, z: Picture(x) \wedge BoaConstrictor(y) \wedge Contain(x, y)$
$\wedge Animal(z) \wedge Swallow(y, z)$

Note that the expression "a picture of X" is explicitly shown as "a picture containing X" in (8) to use the predicate "contain" as a function. The Vietnamese sentence can also be encoded in FOPC notation as follows:

(9) $\exists x, y, z: Trăn(y) \wedge Vẽ(x, y) \wedge Thú(z) \wedge Nuốt(y, z)$

which roughly translates to:

(10) $\exists x, y, z: BoaConstrictor(y) \wedge Draw(x, y) \wedge Animal(z) \wedge Swallow(y, z)$

As seen above, we are unable to overcome syntactic differences using FOPC (a syntax-dependent MR): i.e., $Picture(x) \wedge Contain(x, y)$ should be semantically equivalent to $Draw(x, y)$. This results in different representations for inputs that mean the same thing. Ideally, we would like a way to overlook the differences in their syntactic forms.

The English sentence can be more closely translated into Vietnamese as "Đó là một bức vẽ có một con trăn đang nuốt một con thú" (i.e., "That was one picture having one boa constricting swallowing one animal"). However, the reality is that for translated corpora, there will always be a difference in the manner of expressing an idea, so word-by-word translation is never to be expected. This makes a stronger case for syntax-independence: if we only focus on extracting the literal meanings for two

sentences above, they should end up with the same logical form. On the other hand, if we choose to depend our semantic parsing on syntax, we will encounter these problems:

- To express the concept of "drawing", the English text uses the noun "a picture" while the Vietnamese text uses the verb "draw". "It" in the Vietnamese sentence in fact also refers to the picture itself, and the verb "draw" signifies that "it" refers to a picture. Below is another Vietnamese translation which makes it clear that "Nó" – *"It"* in sentence (6) refers to the picture and not an agent (i.e., the artist):

  (11) Bức tranh vẽ hình một con trăn …
  CLF picture draw image one CLF boa constrictor …

  In (11), the noun that precedes "draw" is "picture" instead of an agent that draws the picture. We see here an example of a concept being expressed differently in two different languages.
- "was" in the English sentence is a copula verb, which we do not see in the Vietnamese sentence.
- Determiners (i.e., "a") are naively translated as numbers (i.e., "một" – *"one"*).
- A verb in its inflected form (i.e., "swallowing") is expressed as two words in Vietnamese (i.e., "đang nuốt").
- The animal classifier "con" (in front of "trăn" and "thú") in sentence (6) does not have a corresponding translation in the Leipzig gloss (7).
- "swallowing" can be either a verb in present continuous tense or part of a participial phrase. In the Vietnamese sentence, it could be misleadingly understood as the former with the function word "đang" prefixed to show tense information, while in fact it is a participial phrase.

These complications with the syntax-driven approach will be categorized and analyzed in the following part.

3.3. Challenges with the Syntax-Driven Approach

This section categorizes and gives background on the language-specific phenomena that bring about the challenges above.

Challenge 1: Classifiers

A classifier is a word or a morpheme that accompanies a noun to give more information about it, hence classifying the noun. This word category does not exist in English. Vietnamese, on the other hand, makes heavy use of classifiers since the fact that it is isolating does not allow inflectional morphemes to be added to indicate a word's properties. In Vietnamese, a classifier immediately precedes a noun. Most pairs of classifier and noun are fixed expressions. Classifiers convey even more information about their following nouns than inflectional morphemes and determiners can, including number, gender, object/animal/person, age, shape and form, and so on.

Challenge 2: Adverbs

This challenge does not appear in the example given, but it exists for the semantic parsing task. As noted above, it is not possible to add suffix to a Vietnamese word to change its category because only one morpheme is allowed per word. Adverbs are therefore often interpreted based on the context and its position in the sentence. Sometimes, for clarity, Vietnamese uses "in a(n) ADJ manner" to indicate an adverbial phrase. Then, the question is how to tell adverbs from adjectives. If our analysis is driven by syntax, we can take into account the word's position in the sentence.

Challenge 3: No to be *before Adjectives*

Unlike English or French, Vietnamese does not require *to be* (i.e., a copula) before adjectives. This use occurs in some cases, for example: "Nó là thật" = "It is true". However, such cases are rare, often take place for emphasis purposes, and can be ungrammatical if not used correctly. This language specificity may be a challenge when we rely on English to parse meaning for a Vietnamese sentence, depending on

how the parsing is done. If the Vietnamese sentence's meaning is directly parsed using an English-based MR (as done in Ha and Nguyen's work in 2019), the process of which is called *in-language annotation*, then the MR scheme needs to be modified to accommodate such syntactic differences. The question is whether we want to do that which will increase the complexity for our model. The next challenge is of the same nature, i.e., requiring some modification to the original scheme to adapt to some language-specific characteristics of Vietnamese.

Challenge 4: Tenses

To specify tenses, Vietnamese also needs to rely on another word category called *function word*. This category is as simplified as possible, limiting to only three tenses: "đã" (past), "đang" (present), "sẽ" (future). Perfect tenses in Vietnamese are not as distinctly defined as those in English. It is often realized by adding adverbs such as "rồi" (already) after the verb.

In the subsequent parts of this chapter, we are going to evaluate each MR based on how well it handles these four challenges. Discussions of the MRs will also interweave with the analysis of each phenomenon, giving another perspective on each MR specifically when it is applied to Vietnamese.

3.4. Modern MRs for Cross-Lingual Approach

In this section, we will reuse sentences (5) and (6) and apply each of the three MRs for semantic annotation. For ease of reference, both sentences (along with the Leipzig gloss for the Vietnamese sentence) are reiterated below:

(5) It was a picture of a boa constrictor in the act of swallowing an animal.

(6) Nó vẽ một con trăn đang nuốt một con thú.
It draw one CLF boa constrictor AUX.PRS swallow one CLF animal.

For each MR, we will perform three procedures. First, we will do in-language annotation using our own understanding of the MR, i.e., directly adopt the MR to annotate the Vietnamese sentence (6) instead of relying on its English counterpart

(5). Then, we do cross-lingual annotation, i.e., obtain the MR for the English half of the bitext (the one in a high-resource language, i.e., *a target language*) then match the semantic representations back to the Vietnamese half (the one in a low-resource language, i.e., *a source language*). The most ideal result is that the MRs obtained for the Vietnamese sentence are similar using both in-language and cross-lingual approaches. However, since this result is not expected, the last procedure is to further evaluate each MR based on how well it handles the challenges outlined in the previous section.

3.4.1. Cross-Lingual Approach

The procedure for cross-lingual semantic parsing that we will employ is inspired by Zhang and colleagues (2018), who carried out an almost exact same task with Chinese as the source language and English as the target language. The MRs they used are PredPatt and UDS. To be more specific, their task involved taking a sentence in Chinese and generating a semantic analysis in the chosen MRs for the English translation of the sentence. The Chinese sentence they used for input, its Leipzig gloss and corresponding English translation are provided in Figure 17.

据 报道 ， 在 比洛克西 ， 三十 人 死 于 一 栋 被 风暴 潮 袭击 的 公寓楼

were reported , in Biloxi , 30 people dead in one block PTCP storm surge hit which flats

"*30 people were reported dead in one block of flats which was hit by a storm surge*."

Figure 17. Chinese sentence with Leipzig gloss and English translation (Zhang et al., 2018)

The task's output is the MR of the sentence's English translation, which is a flat representation for PredPatt (Figure 18) and either a graph or a linearized representation for UDS (Figures 19 and 20 respectively).

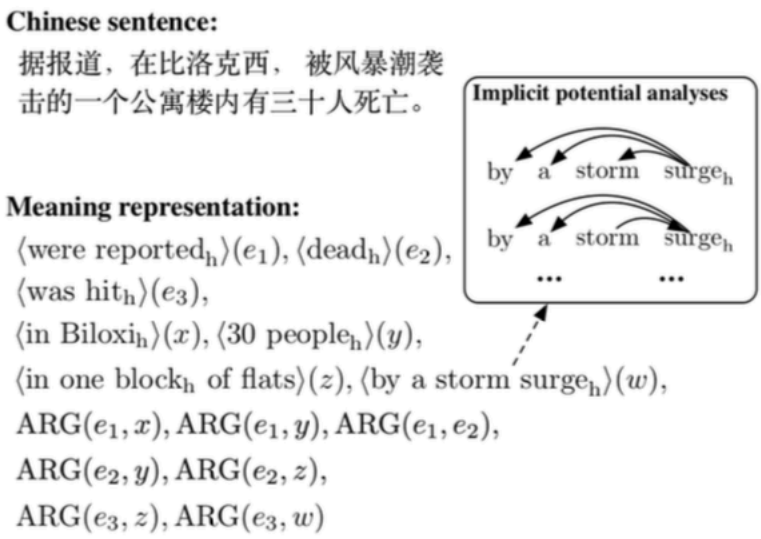

Figure 18. Output MR in target language
after cross-lingual semantic parsing using PredPatt
(Zhang et al., 2018)

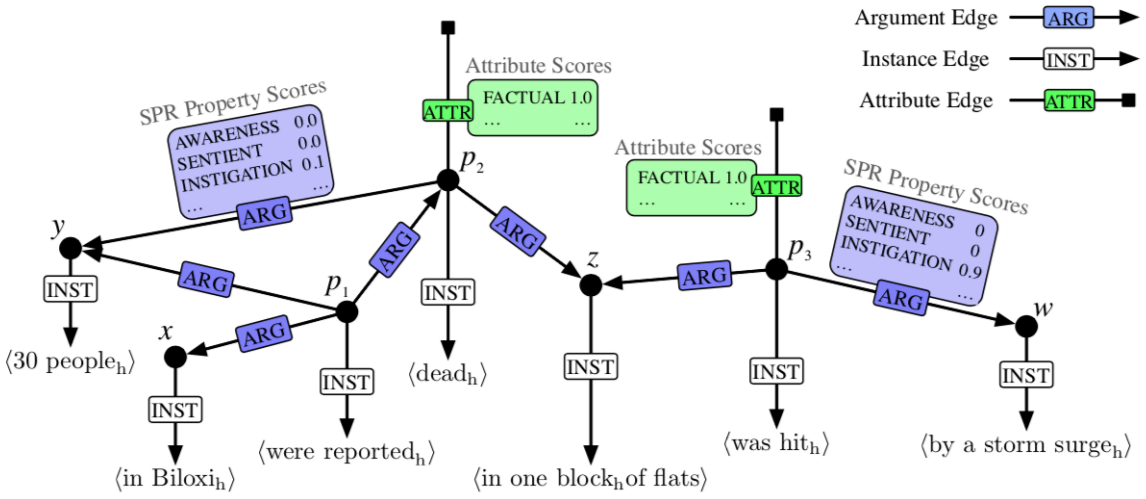

Figure 19. Output MR in target language (graph notation)
after cross-lingual semantic parsing using UDS (Zhang et al., 2018)

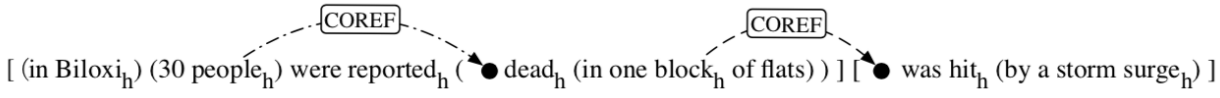

Figure 20. Output MR in target language (linearized notation)
after cross-lingual semantic parsing using UDS (Zhang et al., 2018)

These figures have actually been included and explained in the PredPatt section (2.2.2) and the UDS section (2.2.4).

What we will do differently is, instead of a direct translation of the original sentence, we will utilize a parallel text given in both the source and target languages, which in this case correspond to Vietnamese and English. Moreover, after obtaining the MR for the English half of the bitext, we will match it back to the Vietnamese half, so that the final MR is given on the Vietnamese sentence. (This process will be explained in the next paragraph.) Even though these additions may increase the complexity of our task, they allow us to take advantage of parallel corpora instead of going through an additional translation step and to obtain the MRs in the low-resource language as desired.

The key step for cross-lingual annotation is to align the English-Vietnamese bitext (sentences (5) and (6) above) using the word alignment technique. Bitext word alignment is an NLP task of identifying relationships among translated words (or multi-word units) in a bitext, which is often illustrated in a (bipartite) graph. Figure 21 gives an example of the word alignment for the English-French bitext:

(12) The programe has been implemented.
Le programme a été mis en application.

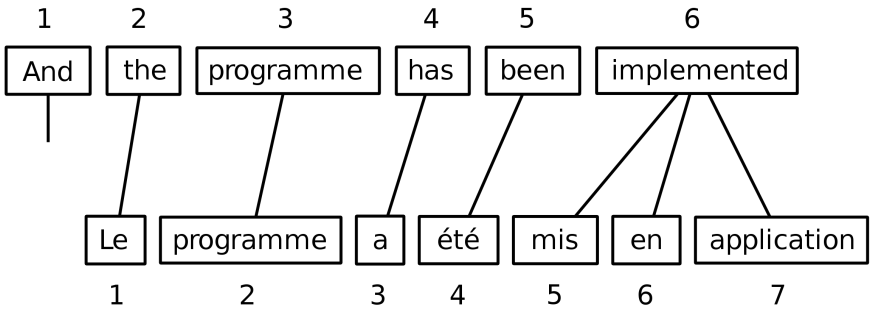

Figure 21. World alignment for the bitext (12)
(https://en.wikipedia.org/wiki/Bitext_word_alignment)

In fact, this word alignment is necessary for us to perform our final step of mapping the sentence's MR from the target language back to the source language. Thus, the

alignment for our bitext is provided in (13). Note that the highlighting only serves to clarify which words or multi-word units correspond to each other.

(13)	It	was a picture of	a	boa constrictor	swallowing	an	animal.
	Nó	vẽ	một con	trăn	đang nuốt	một con	thú.

3.4.2. UDS

In the order presented in the previous section, the first MR we will apply is UDS. The UDS process involves (a) getting the UD parse for the sentence, (b) extracting the predicate and argument roots using PredPatt, and (c) obtaining the linearized representation (we opt for linearized for easier demonstration). Note that we will skip the SPRL step here since it is more efficiently done using a statistical model with SPR data for all English words readily available on Decomp's website: http://decomp.io/projects/semantic-proto-roles/.

Procedure 1: In-Language Annotation

First, we will use our understanding of UDS to directly annotate the Vietnamese sentence (6). Its UD parse (in the CoNLL-U format, detailed documentation at https://universaldependencies.org/format.html) is given below:

order	*token*	*relation*	*translation of relation*
1	Nó	2:nsubj	nsubj(Nó, **vẽ**)
2	vẽ	0:root	root(vẽ, None)
3	một	5:nmod	nmod(một, trăn)
4	con	3:clf	clf(con, một)
5	trăn	7:nsubj	nsubj(trăn, **nuốt**)
6	đang	7:aux	aux(đang, nuốt)
7	nuốt	2:ccomp	ccomp(nuốt, **vẽ**)
8	một	10:nmod	nmod(một, thú)
9	con	8:clf	clf(con, một)
10	thú	7:obj	nobj(thú, **nuốt**)

To briefly explain this format, the first column is the order of the token within the sentence, the second is the token itself, the third is the dependency relation it has to another token, and the last column shows the relation in a more familiar form: $relation(dependent, governor)$. On a side note, the relation $clf(con, một)$ indicates that the classifier "con" is treated as a functional dependent of the numeral "một". This is because syntactically classifiers group with the numerals rather than the nouns, as specified at https://universaldependencies.org/u/dep/clf.html. This observation actually applies to Vietnamese, an example of which is given below:

(14) Trong tất cả những bức tranh này, tôi thích ba bức.
In all CLF picture this, I like three CLF.
"In all of these pictures, I like three (of them)."

Now that we have the UD parse, we will use PredPatt's process to extract the predicate-argument relations. Note that PredPatt is actually a software, but the process will be done manually here using our own knowledge of the system since the software does not handle parses in Vietnamese. Given the relations above, the extracted predicate roots are "vẽ" – *"draw"* and "nuốt" – *"swallow"* (as bolded). Then, each argument to a predicate along with its dependent tokens form an argument phrase, so the argument phrases include "Nó" – *"It"*, "một con trăn" – *"one CLF boa constrictor"*, and "một con thú" – *"one CLF animal"*. After PredPatt, the UDS linearized representation of the sentence is provided below:

(15) [(Nó) vẽ [(một con trăn$_h$) đang nuốt$_h$ (một con thú$_h$)]].

In reality, we could only do in-language annotation if we manually annotated every sentence, which defeats our goal of overcoming scarce resources. To be more practical, we explore the cross-lingual approach in the second procedure, which makes use of a rich-resource language, i.e., English, by relying on a parallel corpus to do semantic annotation for Vietnamese.

Procedure 2: Cross-Lingual Annotation

In this step, we will apply the same steps above to parse the UDS representation for the English sentence (5). First, we obtain the UD parse for the sentence, which looks like below:

order	*token*	*relation*	*translation of relation*
1	It	4:nsubj	nsubj(It, **picture**)
2	was	4:cop	cop(was, picture)
3	a	4:det	det(a, picture)
4	picture	0:root	root(picture, None)
5	of	9:case	case(of, swallowing)
6	a	7:det	det(a, boa)
7	boa	9:nsubj	nsubj(boa, **swallowing**)
8	constrictor	7:compound	compound(constrictor, boa)
9	swallowing	4:advcl	advcl(swallowing, **picture**)
10	an	11:det	det(an, animal)
11	animal	9:obj	obj(animal, swallowing)

As already mentioned in 2.2.2, PredPatt lifts mark and case tokens out of the argument subtree and appends it to the predicate, so "of" is lifted out of the dependency subtree of "swallowing" (since $case(of, swallowing)$ shows that "of" is originally a case token to "swallowing") and appended to the predicate of "swallowing", which is "picture" as evidenced by the relation $advcl(swallowing, picture)$. From the relations above, we can identify that predicate roots include "picture" and "swallowing". The predicate "picture" in this case, however, is different in that not only is it not a verb but it also has dependent tokens. As a result, we will parse a whole predicate subtree containing "picture" and its dependent tokens, which gives "was a picture of". Thus, the predicates are "was a picture of" an "swallowing". Argument phrases, extracted as before, include: "It", "a boa constrictor", and "an animal". Then, we get the UDS linearized representation for this sentence below:

(16) [(It) was a picture$_h$ of [(a boa$_h$ constrictor) swallowing (an animal$_h$)]].

Now, we do the last step of matching the MR back to the source language by replacing the components (essentially predicates and argument phrases) in this UDS representation with their aligned counterparts in Vietnamese (e.g., replacing "It" with "Nó", "was a picture of" with "vẽ", etc.) according to the bitext (13). This step results in:

(17) [(Nó) vẽ [(một con trănₕ) đang nuốtₕ (một con thúₕ)]].

which is exactly as (15)! Thus, we get the same MR parsing the Vietnamese sentence cross-lingually as if we annotated it with the in-language approach.

In fact, this result was beyond my expectation. To a great extent, it has proven the cross-lingual robustness of UDS even only for parsing this particular sentence, because the sentence in and of itself already contains a lot of complications for the cross-lingual task (as we have seen with the failed attempt of the syntax-driven approach). Knowing this, we will move on to evaluate the MR more holistically based on how it handles the four challenges.

Procedure 3: Performance on Four Challenges

This procedure will investigate whether UDS can cover each of the four Vietnamese syntactic phenomena (introduced in 3.2) and how.

Challenge 1: Classifiers

As we have seen above, UDS accounts for classifiers with the relation $clf()$ in the UD parse. In the PredPatt step, this relation will allow the classifier to be grouped into the subtree whose root is the noun it classifies. Then, in the final UDS representation, the classifier is part of the span whose head is the classified noun. When the cross-lingual is applied, we will not be able to obtain this $clf()$ relation in the UD parse. However, since PredPatt groups all modifiers to a noun (i.e., determiners, classifiers, numerals, etc.) into a span whose head is the noun itself, the annotation for the span in the English half of the bitext will be perfectly matched back to that in the Vietnamese half.

Challenge 2: Adverbs

This challenge is actually not encountered in our example above, but it can be overcome in the cross-lingual approach by bitext word alignment. As long as the adverbs in both halves of the bitext are correctly aligned, UDS will be able to tag it with the dependency relation *advmod*() with respect to the verb or the adjective it modifies. With that, the adverb becomes a dependent token of the corresponding verb/adjective and will belong to its subtree in the PredPatt step. UDS linearized representation will in turn treat it and the modified verb/adjective as an entity.

Challenge 3: No *to be* before Adjectives

Due to the fact that many languages (including Vietnamese) do not have overt copulas (i.e., *to be*), UD parse treats copulas as dependents to nonverbal predicates. A nonverbal predicate is expressed with the *nsubj*() relation, as in the example in Figure 22:

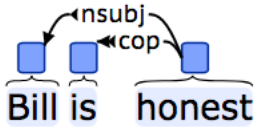

Figure 22. Main relation for the sentence "Bill is honest"[4]

We can see that there exists one main relation *nsubj*(), whose *governor* is "honest" (which is an adjective), i.e., "honest" is the nonverbal predicate in this example. The copula "is" therefore is treated as a dependent token with respect to "honest" through the relation $cop(is, honest)$. PredPatt will process this dependency relation by including the copula as part of the subtree of the nonverbal predicate, and the copula will be part of the predicate in the final representation. Thus, the linearized representation for "Bill is honest" looks as follows:

(18) [(Bill) is honest$_h$].

Using the cross-lingual approach, the copula and adjective in English are aligned with just an adjective in Vietnamese. However, since the copula is part of the

[4] https://universaldependencies.org/u/dep/cop.html

predicate (whose head is the adjective) in the final representation, we will be able to match both of them back to the adjective in the Vietnamese half of the bitext and annotating that adjective to be head of the predicate span. For example, suppose the Vietnamese counterpart sentence of (18) is "Bill chân thật" – *"Bill honest"*, then we have the following word alignment:

(19) Bill is honest.
Bill chân thật.

The MR in (18) can be matched back to the Vietnamese half of the bitext (19) as:

(20) [(Bill) chân $thật_h$].

which is exactly the MR we would get if we directly annotated the Vietnamese sentence in (19).

Challenge 4: Tenses

In the UD parse, function words for tense information are annotated with the relation *aux*() with respect to the verb, e.g., "has done" is annotated as $aux(has, done)$, in which "done" is considered the main verb and "has" indicates the present perfect tense. The fact that the *aux*() dependency is eventually grouped together with the main verb to become a predicate and that the present participle in a participial phrase also acts as a predicate in that phrase helps overcome the potential difference in semantic annotation if confusion arises between the two (the challenge of which has already been explained in 3.2).

3.4.3. AMR

Next, we will apply AMR to parse the MR for sentence (6), again, using both the in-language and cross-lingual approaches as done with UDS.

Procedure 1: In-Language Annotation

Even though we can obtain the AMR for the Vietnamese sentence in one single step, we will start with the logic format and then build the PENMAN notation based on it to show a more procedural and self-explanatory way of obtaining the final

representation. First, we will identify $instance()$ and $arg()$ functions and represent them in the logic format:

(21)

$$\exists\, n, v, t, n2, t2: instance(n, nó) \wedge instance(v, vẽ\_01) \wedge instance(t, trăn) \wedge instance(n2, nuốt\_01) \wedge instance(t2, thú) \wedge arg0(v, n) \wedge arg1(v, t) \wedge arg0(n2, t) \wedge arg1(n2, t2)$$

The $arg()$ relations above indicate two predicates: "vẽ-01" (*v*) – *"draw"* and "nuốt-01" (*n2*) – *"swallow"* (again, the "-01" part is only to indicate which sense of the word the notation refers to). The arguments of "vẽ" include "nó" (*n*) – *"It"* and "trăn" (*t*) – *"boa constrictor"*, and the arguments of "nuốt" include "trăn" and "thú" (*t2*) – *"animal"*. It is worth noting that numerals and classifiers are omitted since AMR drops these categories in its semantic representations (e.g., "một con trăn" – *"one CLF boa constrictor"* is represented as "trăn" – *"boa constrictor"* only). From the logic annotation (21), the PENMAN format is obtained as follows:

(22) (v / vẽ-01
:ARG0 (n / nó)
:ARG1 (t / trăn
:ARG0-of (n2 / nuốt-01
:ARG1 (t2 / thú))))

Note that *:ARG0-of (n2 / nuốt-01)* indicates an inverse relation that means the root (i.e., "trăn") is *arg0* of "nuốt". Inverse relations such as this are used to handle arguments introduced prior to their related predicates.

Another observation to make is that the frameset "vẽ-01" is different from the frameset "draw-01" in English: *arg0* is actually the picture itself, while *arg0* of "draw-01" is the artist, i.e., the person who draws *arg1*. More importantly, there is no relation in English equivalent to "vẽ-01", thus we have to create a relation "vẽ-01" that is specific to Vietnamese. In fact, the case that additional word senses exist in a language other than English can happen for a lot more words, thus AMR's bias towards English will incur more work when it is applied to another language.

From the observation above, a model using AMR will consequently require reconstructing a set of relations for Vietnamese, which is exactly part of Ha and Nguyen's project (2019) to build an MR for Vietnamese based on AMR. However, we want to reduce this effort, so we will incorporate bitext word alignment to experiment with the cross-lingual approach in the next procedure.

Procedure 2: Cross-Lingual Approach

For the cross-lingual approach, instead of going through the logic annotation, we take the PENMAN notation for the English sentence (5) directly from the AMR-annotated *The Little Prince* corpus:

(23) (p / picture
:domain (i / it)
:topic (b2 / boa
:mod (c2 / constrictor)
:ARG0-of (s / swallow-01
:ARG1 (a / animal))))

The relation *:domain* is used to represent copulas, with the idea that the predicate (i.e., "picture" in this case) is within the domain of the argument (i.e., "it"). For example, *:domain* is also used to represent sentences such as "She is smart":

(24) (s / smart
:domain (s2 / she))

in which "smart" is considered the predicate and "she" is its single argument. Note that "she" is not annotated with the *:arg* relation with respect to "smart" because "smart" is not a frameset in AMR and thus does not have *arg0*, *arg1*, etc. defined. The relation *:topic* indicates that the argument (i.e., "boa" in this case) is the topic of the predicate (i.e., "picture"), which roughly means that "picture" is about or of "boa". The relation *:mod* indicates that "constrictor" is a modifier of "boa" even though it technically is not. However, this is just how AMR handles multi-word units and does not affect how the meaning is interpreted in any way.

From (24), we match the representation back to the Vietnamese half of the bitext (13) and get:

(25) (v / vẽ-01
:domain (n / nó)
:topic (t / trăn
:ARG0-of (n2 / nuốt-01
:ARG1 (t2 / thú))))

The *:mod* relation is absent from (25) since the multi-word unit "boa constrictor" is matched to a single word, "trăn", in Vietnamese. This representation is actually pretty close to Ha and Nguyen (2019)'s annotation of the sentence, which is:

(26) (v / vẽ-01
:domain (n / nó)
:topic (t / trăn
:ARG0-of (n2 / nuốt-01
:tense (đ / đang)
:ARG1 (t2 / thú))))

The only difference is that Ha and Nguyen augmented the annotation scheme by adding a relation *:tense* to keep track of tense information (2019), i.e., "đang" for the present continuous tense in this case.

However, the representation (25) obtained from the cross-lingual approach is not the same as (22), which we got by directly annotating the sentence in Vietnamese. This difference in fact stems from how specifically AMR handles expressions in English, e.g., representing "of" in "a picture of" with the *:topic* relation, while other languages may very well not have the same way of expressing the concept of "drawing". Thus, even though AMR's central idea of using framesets and a defined set of relations serves the goal of being syntax-independent, it poses a difficulty in applying the MR scheme to non-English languages.

Procedure 3: Performance on Four Challenges

In addition to seeing how AMR does in both approaches for semantic annotation, we want to evaluate whether it can overcome the four challenges posed at the beginning of this chapter.

Challenge 1: Classifiers

AMR chooses to drop word categories that do not impact the meaning of the sentence, which include classifiers (among others such as numerals, determiners, and so on). Thus, if a determiner in English (e.g., "a") corresponds to a numeral and a classifier in Vietnamese (e.g., "một con" – *"one CLF"*) for example, all of them will be dropped in the AMR of the sentence they belong to, allowing for matching MRs when we do the cross-lingual approach.

Challenge 2: Adverbs

Adverbs get stemmed to their adjective forms and are annotated with the *:manner* relation in AMR[5]. As mentioned in 3.2.3, Vietnamese uses "in a(n) ADJ manner" to indicate an adverbial phrase because it cannot add a suffix to an adjective (due to the language's single-morpheme characteristic). Let us consider some bitext whose English half contains the adverb "quickly", which corresponds to "nhanh" in the Vietnamese half. The word "nhanh" can either be an adjective or an adverb, but our original challenge is that there is no way to tell the difference that does not rely on the word's relative position in the sentence. Now, with the cross-lingual approach, the English adverb will be annotated as ":manner (q / quick)", which matches back to Vietnamese as ":manner (n / nhanh)", immediately indicating that "nhanh" is an adverb and also conveniently portraying the expression "in a(n) ADJ manner" for an adverbial phrase in Vietnamese.

[5] https://github.com/amrisi/amr-guidelines/blob/master/amr.md

Challenge 3: No *to be* before Adjectives

As already encountered in the cross-lingual procedure, copulas (i.e., *to be*) are annotated with the *:domain* relation, indicating that the adjective or noun that follows is within the domain (i.e., is a quality or a part) of the preceding noun. This annotation matches back to the Vietnamese half of a bitext perfectly since the only tokens that actually are included in the MR are the subject and the adjective or noun that follows the copula (but not the copula), both of which should be present in the Vietnamese sentence.

Challenge 4: Tenses

AMR opts not to annotate tenses, but tense information can be accounted for by adding a relation *:tense*, as done in Ha and Nguyen's project (2019) and demonstrated above.

3.4.4. UCCA

In this section, we will employ the last MR scheme, UCCA, to parse the semantics for the Vietnamese sentence both in-language and cross-linguistically. Then, like the two previous MRs, UCCA will also be evaluated on how it handles the four syntactic phenomena specific to Vietnamese.

Procedure 1: In-Language Annotation

Employing the same procedures as UDS and AMR, we first start by attempting in-language annotation for sentence (6). Using our own knowledge and understanding of UCCA, we obtain the following linearized representation:

(27) Nó$_A$ vẽ$_S$ [[một$_E$ con$_E$ trăn$_C$]$_A$ đang$_D$ nuốt$_P$ [một$_E$ con$_E$ thú$_C$]$_A$]$_A$.

Picturing the UCCA graph that (27) represents, we have the first level of edges containing Participant (*A*), State (*S*), Participant (*A*) edges that lead to the terminal and non-terminal units "Nó" – *"It"*, "vẽ" – *"draw"*, and "một con trăn đang nuốt một con thú" – *"one CLF boa constrictor AUX.PRS swallow one CLF animal"* respectively. Note that "vẽ" is annotated as a State rather than a Process (*P*) since, in the Vietnamese

sentence, the word introduces a description of the picture (which the pronoun "Nó" refers to) instead of being an action of "Nó" (again, because "Nó" here refers to the picture, not an agent that draws the picture). Next, the second level of edges will further break down "một con trăn đang nuốt một con thú" into four units: "một con trăn" – *"one CLF boa constrictor"*, "đang" – *"AUX.PRS"*, "nuốt" – *"swallow"*, and "một con thú" – "*one CLF animal"*, whose roles in the superordinate relation correspond to Participant, Adverbial (D), Process (P), and Participant. Note that the auxiliary for present continuous tense (i.e., "đang") is annotated with an Adverbial because UCCA considers auxiliaries as semantically modifying verbs, which classifies them under Adverbial relations. The last level of edges breaks down the non-terminal units "một con trăn" and "một con thú" each into three terminal units, which get the roles Elaborator (E), Elaborator (E), and Center (C) correspondingly. Numerals (e.g., "một" – *"one"*) and classifiers (e.g., "con") are all considered Elaborators since they provide more information (e.g., number, object/animal/human, gender, etc.) about a specific entity, which are whatever nouns that immediately follow (i.e., "trăn" – *"boa constrictor"* and "thú" – *"animal"* in our example).

Procedure 2: Cross-Lingual Annotation

Now, we will attempt the cross-lingual approach with UCCA. This time, to annotate the English half of the bitext, i.e., sentence (5), we will actually use the web interface[6] of a transition-based UCCA parser called TUPA (Hershcovich et al., 2017, 2018) to get the graph representation for the sentence, displayed in Figure 23:

[6] https://www.cse.huji.ac.il/~danielh/ucca/

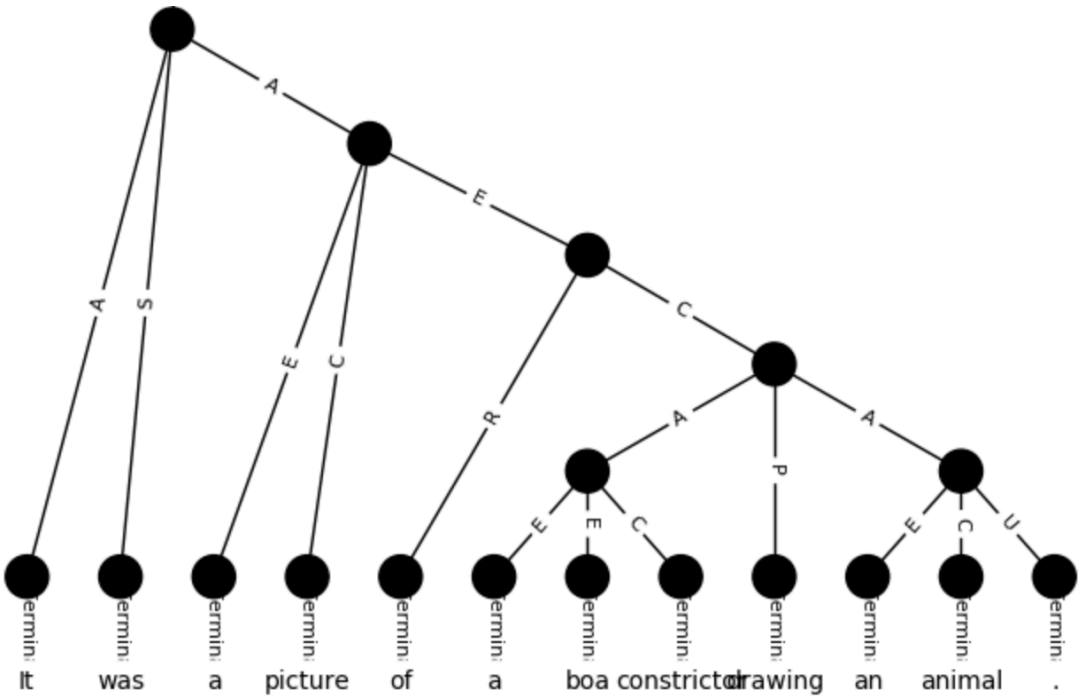

Figure 23. UCCA annotation graph for sentence (5)

We can obtain the linearized format for the graph above as:

(28) It$_A$ was$_S$ [a$_E$ picture$_C$ [of$_R$ [a$_E$ boa$_E$ constrictor$_C$]$_A$ swallowing$_P$ [an$_E$ animal$_C$]$_A$]$_E$]$_A$.

Interestingly, still following UCCA annotation guidelines[7], we can get yet another parse for the sentence. The guidelines introduce the case of *classifier construction*, in which a Center (*C*) specifies a particular instance and the Elaborator (E) specifies the instance's type. To give an example for this abstract case, the phrase "the story of a young girl sentenced to death" would be annotated with UCCA as follows:

(29) [the$_E$ story$_C$ of$_R$]$_E$ [[a$_E$ young$_E$ girl$_C$]$_A$ sentenced$_P$ [to$_R$ death$_C$]$_A$]$_C$.

In (29), the instance specified as a Center here is "a young girl sentenced to death" and the type of this instance is "the story", which is considered an Elaborator. The structure of the phrase represented in (29) is very similar to that of the phrase "a picture of a boa constrictor swallowing an animal". Thus, the whole English sentence we are parsing can be alternatively represented as:

(30) It$_A$ was$_S$ [[a picture of]$_E$ [[a boa constrictor]$_A$ [swallowing]$_P$ [an animal]$_A$]$_C$]$_A$.

[7] https://universalconceptualcognitiveannotation.github.io/

The representation in (30) indicates that "a picture" is the type of the instance "a boa constrictor swallowing an animal", thus the roles assigned to the two units respectively are Elaborator (*E*) and Center (*C*). This situation demonstrates one of the potential ambiguities in considering the same semantics of a sentence using UCCA's annotation scheme. The ambiguities may stem from the fact that the elements (i.e., categories) in this MR scheme are cognitively motivated and not defined with concrete rules, thus creating room for different possible annotations for the same semantic units. All this discussion aside, we will stick with the representation generated by TUPA and linearized in (28).

Now, as we attempt to map the UCCA linearized representation in (28) back to the Vietnamese half of the bitext, we encounter a problem that "was a picture of" is not a single entity with one category, making it impossible to assign a category to the phrase's Vietnamese counterpart, i.e., "vẽ". The alternative parse in (30) comes very close but "was" does not have the same category as "a picture of", thus still not allowing us to perfectly match the semantic representation back to the Vietnamese sentence either.

Procedure 3: Performance on Four Challenges

We will move on to investigating how UCCA can handle the four challenges introduced at the beginning of this chapter in order to gain a more comprehensive evaluation of the MR scheme.

Challenge 1: Classifiers

Since there are no classifiers in English, the classifiers in the Vietnamese half of a bitext will not receive a separate annotation if we use the cross-lingual approach. (The in-language approach will annotate classifiers as Elaborator (*E*), as seen above). However, classifiers are most of the time aligned with determiners in the English half, which are annotated as Elaborators, which means the classifiers will receive the role Elaborator when the annotation is matched back to the Vietnamese half. For example, "một con" (a numeral + a classifier) is aligned with the determiner "a" in

the English half of our bitext (13). Since "a" is annotated as an Elaborator, "một con" in turn will also be assigned the category *E* for Elaborator, which is exactly what we got when we annotated the Vietnamese sentence directly instead of relying on its English counterpart. UCCA therefore correctly handles classifiers even when it is applied cross-lingually.

Challenge 2: Adverbs

UCCA handles adverbs by recognizing the role they play in relation to the Process and classifying them as *D* (for Adverbial). With bitext word align, this annotation will be matched back to the corresponding adverb in Vietnamese without requiring us to identify whether the Vietnamese word is an adjective or an adverb.

Challenge 3: No *to be* before Adjectives

For English, UCCA represents a copula before an adjective by regarding the copula with the adjective as a static relation (i.e., using *S* to annotate both words as a single entity), or by labeling the copula as *F* (standing for Function, which is only required by the structural pattern it appears in, which applies to this case because *to be* before adjectives is required in English grammar but not in Vietnamese, thus is a structural pattern) and the adjective as the main relation *S* (for State). In the first way, bitext word alignment will match the annotation *S* back to the adjective in the Vietnamese half (since a copula + an adjective in English will be aligned with only an adjective in Vietnamese) as desired. In the second way, if the copula in the English half is marked as *F*, we can just ignore the copula and only match the *S* relation of the adjective back to the adjective in Vietnamese, also yielding a correct annotation for the Vietnamese half.

Challenge 4: Tenses

Tense auxiliaries can be viewed as modifying verbs, which UCCA annotates as Adverbials (*D*). However, since English does not have certain tense auxiliaries like Vietnamese does (e.g., English does not have an equivalent function word for "đang"

which indicates the present continuous) but in which case uses additional morphemes instead, we can separate the verb form into its stem and the morpheme for further annotation. For example, "swallowing" can be divided into "swallow" and "-ing", which are in turn annotated as a Process (*P*) and an Adverbial (*D*) respectively. UCCA in fact allows for terminal units to be morphemes for more specific annotation. This will perfectly match back to the two-word unit "đang nuốt" in Vietnamese (i.e., "đang" will receive the annotation *D* for Adverbial like its counterpart "-ing" and "nuốt" will receive *P* for Process like "swallow") if we do cross-lingual annotation. However, as we have seen above, the TUPA parser does not automatically split an inflected verb into its stem and its added morpheme. Cross-lingual annotation will therefore require additional steps of identifying which inflected verbs in the English half of a bitext match to two words (an auxiliary + a verb) in the Vietnamese half and customizing the parser to handle splitting for these verbs. A simpler way we could do this is to keep the inflected verb as is. With bitext word alignment, the corresponding two-word verb in Vietnamese will be regarded as a single entity and annotated with the same category. More specifically, if we keep "swallowing" as a complete word without stemming it and assign to it the Process (*P*) relation, then its Vietnamese counterpart in the bitext, i.e., "đang nuốt", will be treated as a single entity and annotated with Process, and there will be no further separate annotations for "đang" and "nuốt" (i.e., "đang nuốt" will be a terminal unit).

4. Discussion on Cross-Lingual Potential

Thus far, we have studied three MR languages: UDS, AMR, and UCCA, and explored how to each of them can be applied to do cross-lingual semantic parsing for Vietnamese. To evaluate the performances of these MRs, we employ each of them for the task of generating an MR for a Vietnamese sentence, taken from *The Little Prince* corpus, whose English parallel sentence is also given to form an English-Vietnamese bitext to be used in the cross-lingual approach. We first compared the MRs obtained for the Vietnamese sentence through both in-language and cross-lingual approaches to see if they matched. Then, we further assessed and explained how each MR handles

four syntactic phenomena in Vietnamese that posed difficulties for the traditional syntax-driven semantic parsing approach.

From the results in §3, the only MR that was able to produce matching MRs from both the in-language and cross-lingual approaches is UDS. The AMR obtained from the cross-lingual approach differs from our direct annotation on the Vietnamese sentence due to the MR's specific relations that are particularly designed for English. For UCCA, the cross-lingual approach failed to match the MR back to the Vietnamese sentence, which we attributed to its cognitively-motivated categories that could lead to multiple ways of representing the same meaning. However, all three MRs are able to handle the four Vietnamese syntactic phenomena, even though each has a different mechanism to do so.

UDS performs best in the cross-lingual approach, while AMR and UCCA have certain drawbacks to overcome. This final result is interesting since UDS is the only MR out of the three that relies its semantic annotation on syntactic dependencies while the other two aim to be syntax-independent and have their own semantic relations or categories. However, UDS's performance is perhaps due to the fact that it simplifies the backbone of its semantic annotation to predicate-argument structures only and leaves the heavier task of deeper semantic analysis to add-on properties such as SPR or attribute scores. The defined annotation rules employed in the UD parse (i.e., first step of UDS) also guarantees a single way to parse dependencies among tokens, resulting in only one representation per meaning (i.e., only one possible representation when predicate-argument relations are parsed in the PredPatt step), unlike the case of UCCA.

Despite UDS's superior performance in the cross-lingual approach, there currently exists no parallel English-Vietnamese corpus whose English part is annotated in UDS. It is worth noting that there is a UD treebank in Vietnamese, which can be beneficial for training an in-language parser (instead of a cross-lingual one, due to no parallel corpus available). Meanwhile, AMR provides a semantically annotated corpus of *The Little Prince* in English, for which a parallel (unannotated) text in Vietnamese is also available. Similarly, there exists a UCCA-annotated

English corpus of *Twenty Thousand Leagues Under the Seas*, whose Vietnamese (unannotated) version can be easily obtained as well.

5. Next Step: Building a Semantic Parser

The next step for this thesis is to build a comprehensive parser for Vietnamese. This second half involves exploring tools that can be used to automate the cross-lingual semantic parsing task, choosing a dataset to work with, designing a process to train and evaluate the parsing model, and finally implementing the parser. The linguistics study conducted up to this point has provided thorough and valuable background on three of the most robust MRs for the cross-lingual semantic parsing task, which will be crucial in deciding which MR to use in the final implementation and what data is available for training and testing the semantic parser.

5.1. Choice of MR

In previous sections, we analyzed the performance of AMR, UDS, and UCCA and found out that UDS gave the best results. However, what is crucial for building the semantic parser is the available data annotated in an MR language. Specifically, we need an annotated English corpus which has a parallel corpus in Vietnamese (which is not expected have annotations). The word alignments for this bilingual EN-VN corpus should be obtainable with high accuracy. We will discuss what resources are available for each MR to arrive at an informed decision on the final candidate.

UDS has an annotated Vietnamese corpus called the Vietnamese UD Treebank whose texts were extracted from an online Vietnamese daily newspaper (Nguyen et al., 2009). It contains 3,000 sentences (with a total of 43,754 tokens) annotated in UDS (UDS website, 2020). However, a parallel corpus in English for it does not exist, and generating such a corpus will not be a trivial task because it involves first of all translating each of the 3,000 sentences into English.

AMR provides complete annotations for the English corpus of *The Little Prince*, which has an unannotated parallel corpus in Vietnamese. The sentences are translated pretty closely, so it is possible to generate high-accuracy word alignments

for this bilingual corpus. The problem with AMR is that it completely relies on a list of PropBank framesets that are only given in English, so creating AMR annotations for a Vietnamese corpus requires building a similar list of framesets for Vietnamese, which will take a tremendous amount of work since there are 7,312 entries in the English frameset list and one for Vietnamese will require around the same number of entries if not more. The project to construct framesets similar to PropBank for Vietnamese has been carried out by Ha and Nguyen in 2019, but the resulting frameset bank has not been released publicly.

UCCA annotations are created for the first five chapters of *Twenty Thousand Leagues Under the Sea*'s English text (*20K Leagues* for short), the original version of which has also been translated into Vietnamese. UCCA's semantic categories are designed to be applicable across languages, thus they can supposedly cover all words and expressions in any language aside from English.

Ever since its introduction in 2003, UCCA has attracted a lot of research on developing annotation resources for it, notably through public parsing competitions (e.g., SemEval, CoNLL) that accompany top-tier NLP conferences. These efforts have cumulated a wealth of tools for UCCA to streamline the annotation task for different languages, including standard guidelines, an annotation web application (UCCAApp), and seven large annotated corpora in three languages (English, French, and German) with over 30,000 tokens each (UCCA website, 2020). The most valuable tool UCCA has is a parser named TUPA, which was developed by its own creators and can parse the meaning of any English sentence and visualize it in a graph. TUPA allows us more freedom with the choice of corpus to be used, since we can easily obtain UCCA annotations by ourselves using this tool without being restricted to UCCA's already-provided corpora.

From the analyses above, it is clear that our final choice was UCCA, given its portability across languages and abundance in resources that broadens our options for corpora to use.

5.2. TUPA – A Transition-Based Parser for UCCA

One of the main reasons UCCA was chosen is that it has its own parser that can be trained on any language as long as there are UCCA semantic parses in that language for the model to learn, giving us the flexibility to choose our own source of training data. TUPA was developed by Daniel Hershcovich along with the creators of UCCA, Omri Abend and Ari Rappoport, and was first introduced in 2017. It is a transition-based parser which uses a set of transitions and features with an underlying ML model.

According to Hershcovich et al. (2017), transition-based parsers create a parse for a given text incrementally by scanning it from start to end and applying a transition to the parser's state at each step. The next transition is selected by a *classifier* based on features that encode the current state. To give a glimpse of how it actually works, we reproduce the complete transition set in Figure 24.

Before Transition				Transition	After Transition					Condition
Stack	**Buffer**	**Nodes**	**Edges**		**Stack**	**Buffer**	**Nodes**	**Edges**	**Terminal?**	
S	$x \mid B$	V	E	SHIFT	$S \mid x$	B	V	E	$-$	
$S \mid x$	B	V	E	REDUCE	S	B	V	E	$-$	
$S \mid x$	B	V	E	NODE$_X$	$S \mid x$	$y \mid B$	$V \cup \{y\}$	$E \cup \{(y,x)_X\}$	$-$	$x \neq$ root
$S \mid y, x$	B	V	E	LEFT-EDGE$_X$	$S \mid y, x$	B	V	$E \cup \{(x,y)_X\}$	$-$	$\begin{cases} x \notin w_{1:n}, \\ y \neq \text{root}, \\ y \not\rightsquigarrow_G x \end{cases}$
$S \mid x, y$	B	V	E	RIGHT-EDGE$_X$	$S \mid x, y$	B	V	$E \cup \{(x,y)_X\}$	$-$	
$S \mid y, x$	B	V	E	LEFT-REMOTE$_X$	$S \mid y, x$	B	V	$E \cup \{(x,y)^*_X\}$	$-$	
$S \mid x, y$	B	V	E	RIGHT-REMOTE$_X$	$S \mid x, y$	B	V	$E \cup \{(x,y)^*_X\}$	$-$	
$S \mid x, y$	B	V	E	SWAP	$S \mid y$	$x \mid B$	V	E	$-$	$\mathrm{i}(x) < \mathrm{i}(y)$
[root]	$\emptyset$	V	E	FINISH	$\emptyset$	$\emptyset$	V	E	$+$	

Figure 24. The transition set of TUPA (Hershcovich et al., 2017)

During this process, three data structures are involved: a buffer B to store tokens and nodes to be processed, a stack S to push nodes that are being processed, and a graph $G = (V, E, \ell)$ where V denotes the set of nodes, E denotes the set of edges, and $\ell : E \rightarrow L$ is the label function with L representing the set of possible labels (Hershcovich et al., 2017). When the state is marked as *terminal*, G is the final output. Figure 24 involves more details that are not going to be explained here (since it is not the main focus of this thesis), so interested readers should refer to "A Transition-Based Directed Acyclic Graph Parser for UCCA" by Hershcovich et al. (2017) for a better understanding of the workings behind TUPA.

TUPA offers three model architectures (i.e., choices for the classifier): a linear classifier with sparse features (referred to as *sparse*), a feedforward neural network with dense embedding features (also called a *multi-layer perceptron* or MLP), and a network with bidirectional long short-term memory (i.e., BiLSTM) feature representation on top of dense embedding features, which Hershcovich et al. experimented with to investigate the effectiveness of different classifiers in parsing UCCA graphs (2017). As these short descriptions suggest, the choice of classifier determines how features are represented, both of which play an important role in transition-based parsing (Chen and Manning, 2014; Andor et al., 2016; Kiperwasser and Goldberg, 2016).

There are two ways TUPA can be used. One may either run its *pretrained* model, which was trained on the UCCA Wiki corpus[8] containing around 160K tokens, on English sentences saved in TXT files. The other way is to train a model from scratch, specifying one's choice of model type (sparse, MLP, or BiLSTM) and providing training and development data in a given language, and use the trained model (saved in the model files) to parse sentences of that language.

5.3. Use of Bitext Word Alignment

As one of our main goals is to experiment with a cross-lingual approach to generate annotated corpora for low-resource languages, bitext word alignment is chosen as the backbone technique for converting annotated resources from English to Vietnamese. This technique supports many NLP tasks and is particularly indispensable for statistical machine translation (SMT). In 3.4.1, this technique has been defined as "an NLP task of identifying relationships among translated words (or multi-word units) in a bitext" and illustrated through an example, so now we will go briefly into how word alignments are created for a large number of texts.

Alignments can be formed on different levels (ranging from most general to most specific): document, sentence, sub-sentence (e.g., chunk or phrase), and word. While it may take minimal effort to obtain aligned documents since most provided

[8] https://github.com/UniversalConceptualCognitiveAnnotation/UCCA_English-Wiki

bitexts are document-aligned (such as an article in two languages), more specific alignments are much more challenging to create, with the hardest being word alignment, because they require handling nuances in language expressions which are better done with subjective knowledge (e.g., by a human annotator). In reality, the number of documents (and thus words) to align should be significant enough to produce a useful corpus. However, with a lack of human resources especially for lesser-known languages, the task of bitext word alignment needs to be automated using a model.

The two types of models that are most widely used are IBM models and models based on Hidden Markov Model (HMM) (HMM-based models for short). There are currently five official IBM models, each of which is an improvement on the previous. The sixth model is being developed with the goal to incorporate HMM to improve the alignments. According to Och and Ney (2003), IBM Model-4 is the current state of the art.

The process to produce word alignments is summarized in Figure 25.

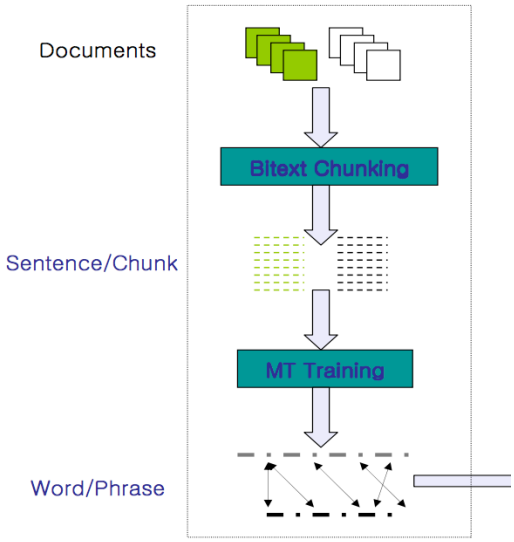

Figure 25. Process behind a bitext alignment model (Deng, 2005)

As shown in Figure 25, this process starts with parallel documents, which are tokenized or segmented into space-delimited token sequences during preprocessing. At the bitext chunking stage, sentence or chunk pairs are derived from these tokens,

which then act as training material for the machine translation (MT) training stage to build statistical word and phrase alignment models (Deng, 2005). These alignment models produce matchings of words or phrases within a bitext (demonstrated with black arrows in Figure 25) which are predicted using probability (hence the term *statistical*).

We conclude this subsection by showing an example of word alignments for an EN-VN bitext:

(31) I had rarely seen him so animated.

(32) Ít khi tôi thấy hắn sôi nổi như thế.

The resulting word alignments are [1-3], [3-1,2], [4-4], [5-5], [6-8,9], [7-6,7], [8-10], which are collectively visualized in Figure 26.

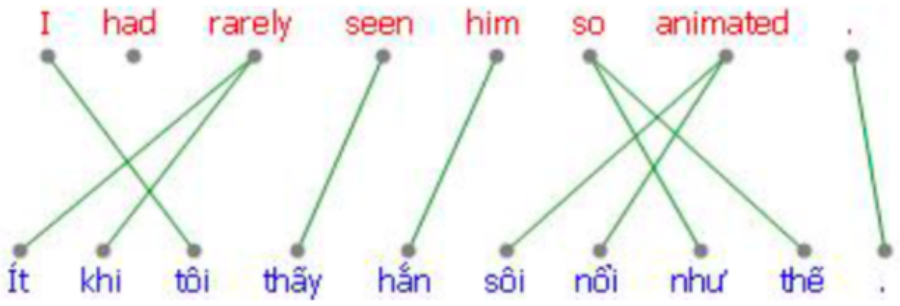

Figure 26. Word alignments for a bilingual sentence pair
(Ngo et al., 2013)

6. Methodology

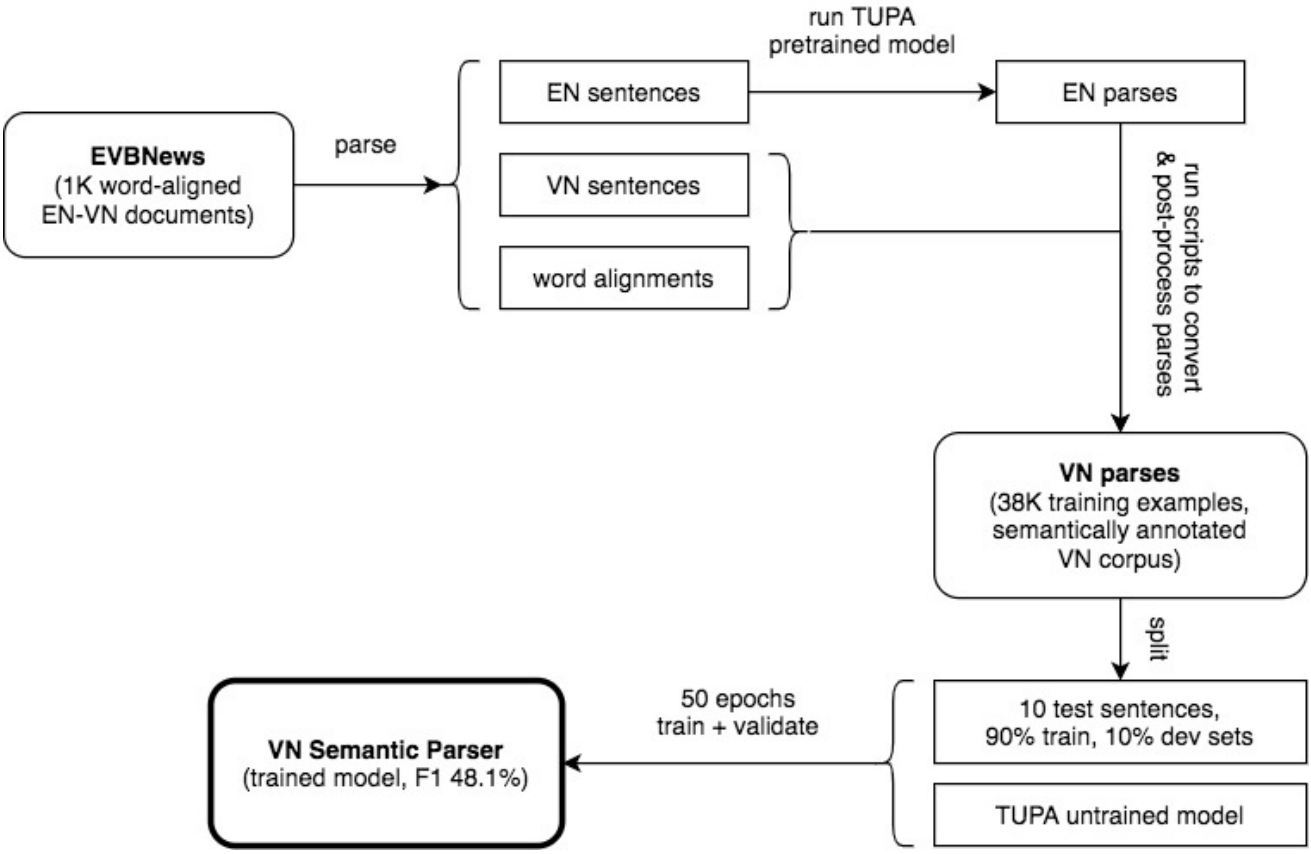

Figure 27. Flow diagram of this research

The methodology used to build our semantic parser is summarized in Figure 27. Further details regarding each step are provided in the following subsections.

6.1. Source Data

For our semantic parser, we did not generate word alignments from scratch but took advantage of an English-Vietnamese (EN-VN) bilingual corpus, EVBNews, which had already been aligned on the word level. This corpus is the News portion of the English-Vietnamese Bilingual Corpus (EVBCorpus), a large-scale project by Ngo et al. in 2013 to study tasks in comparative linguistics and MT. EVBNews contains 1,000 word-aligned parallel documents that are originally articles from a multitude of news sources and range over a variety of topics (Ngo et al., 2013; Ngo and Winiwarter, 2012). The process to create alignments for this corpus was overall

similar to the one explained in 5.3, except that the annotators corrected the final alignments manually to ensure it is a gold-standard corpus (Ngo et al., 2013).

According to Ngo et al. (2013), the bilingual news articles are annotated with both sure alignments and possible alignments, which help evaluate the model with the Alignment Error Rate (AER) (Och and Ney, 2003). In 1,000 aligned documents, there are 447,906 sure alignments, accounting for 80% of 560,215 possible alignments in total. These sure alignments mainly come from nouns, verbs, adverbs, and adjectives, which generally are meaningful words in sentences (in contrast to function words, for example). Most of the remaining 20% are prepositions in both English and Vietnamese (Ngo et al., 2013). The statistics were reported by the authors as follows:

	English	Vietnamese
Files	1,000	1,000
Sentences	45,531	45,531
Words	740,534	832,441
Sure Alignments	447,906	447,906
Possible Alignments	560,215	560,215
Words in Alignments	654,060	768,031

Figure 28. Statistics of EVBNews from authors (Ngo et al., 2013)

For comparison, the statistics recorded after we processed the EVBNews corpus for the semantic parser are shown in Table 3:

	English	**Vietnamese**
Documents	1,000	1,000
Sentences	45,308	45,308
Words	779,242	1,105,277
Annotated Sentences	45,287	38,160

Table 3. Statistics of EVBNews recorded from our preprocessing

The discrepancy in word counts possibly results from different criteria for counting words. The non-matching numbers of sentences are unaccounted for at this point but

should not affect our task moving forward. Since this bilingual corpus was produced with sentence-by-sentence translations, the numbers of English and Vietnamese sentences are both 45,308 as reported in Table 3.

The 1,000 documents are in Standard Generalized Markup Language (SGML) format, whose filenames range from *N0001.sgml* to *N1000.sgml*. Each file contains a bilingual article separated into pairs of EN-VN sentences. In particular, each sentence is presented in a block with three main lines: the first being its EN version, the second its VN version, and the third containing word alignments for the bitext. Figure 29 shows how a sentence is presented in one of these files.

```
<spair id='1'>
<s id='en1'>What is a Fenqing ?</s>
<s id='vn1'>Fenqing là gì ?</s>
<a id='ev1'>1-3;2-2;4-1;</a>
</spair>
```

Figure 29. Example sentence in a SGML file of EVBNews

At this point, it is also worth noting that this corpus is not annotated semantically. In the next step, we would leverage UCCA's parser to generate annotations for it, which would then become our training data.

6.2. Process to Create Training Data

1. Generated UCCA parses for all EN sentences using the pre-trained TUPA model, one parse per sentence.

Even though UCCA is a powerful semantic annotation scheme in that it can annotate cross-sentence meanings (in forms of paragraphs/passages), the common practice in semantic parsing is to parse meanings from individual sentences (Hershcovich et al., 2017). Furthermore, since the bilingual documents in EVBNews are not separated by paragraph, we would have to input the entire document as a passage, which would affect the performance of the model. Therefore, we proceeded with the following steps instead.

First, we wrote a script to parse each single-language sentence in every file and save it to a TXT file. The filename is the sequential number of a sentence relative to all sentences in the corpus (not the document), which is henceforth called the sentence ID, so the last sentence of the 1000th document would be stored in file *45308.txt*. Two folders were created to store EN and VN sentences separately. For example, Figure 29 is the first sentence of the very first SGML document in EVBNews, thus its sentence ID is 1. After parsing, file *1.txt* in the folder for VN sentences contains "Fenqing là gì?". Its parallel EN text, "What is a Fenqing?", is saved in file *1.txt* in the EN sentence folder.

Next, we used TUPA's *pretrained* model to generate annotations for the EN sentences. This pretrained model is reported to achieve an overall accuracy of 73.5% (Hershcovich et al., 2017). It takes as input a TXT file generated in our previous step, parses the sentence's meaning, and produces an XML file that stores this semantic parse. The content of this XML file looks as follows:

```
<root annotationID="0" passageID="1_0">
  <attributes />
  <extra format="ucca" />
  <layer layerID="0">
    <attributes />
    <extra doc="[[[5931147113347055926, 5865838185239622912, 4808651922106831370, 95,
0,  2,  404,  1,  10887629174180191697,  14825646868312541310,  14949295185858420483],
[3411606890003347522,  10382539506755952630,  13927759927860985106,  100,  0,  2,
8206900633647566924, 0, 4370460163704169311, 5097672513440128799, 3411606890003347522],
[11901859001352538922, 11901859001352538922, 15267657372422890137, 90, 0, 2, 415, 1,
11123243248953317070,  11901859001352538922,  11901859001352538922],  [null,  null,
15308085513773655218, 92, 380, 3, 429, -2, 16072095006890171862, 9148987913859918431,
7679303661980345986], [8205403955989537350, 8205403955989537350, 12646065887601541794,
97, 0, 2, 445, -3, 8205403955989537350, 8205403955989537350, 8205403955989537350]]]" />
    <node ID="0.1" type="Word">
      <attributes paragraph="1" paragraph_position="1" text="What" />
    </node>
    <node ID="0.2" type="Word">
      <attributes paragraph="1" paragraph_position="2" text="is" />
    </node>
    <node ID="0.3" type="Word">
      <attributes paragraph="1" paragraph_position="3" text="a" />
    </node>
    <node ID="0.4" type="Word">
      <attributes paragraph="1" paragraph_position="4" text="Fenqing" />
```

```
    </node>
    <node ID="0.5" type="Punctuation">
      <attributes paragraph="1" paragraph_position="5" text="?" />
    </node>
  </layer>
  <layer layerID="1">
    <attributes />
    <node ID="1.1" type="FN">
      <attributes />
      <edge toID="1.2" type="H">
        <attributes />
        <category tag="H" />
      </edge>
    </node>
    <node ID="1.2" type="FN">
      <attributes />
      <edge toID="1.3" type="A">
        <attributes />
        <category tag="A" />
      </edge>
      <edge toID="1.4" type="S">
        <attributes />
        <category tag="S" />
      </edge>
      <edge toID="1.5" type="A">
        <attributes />
        <category tag="A" />
      </edge>
    </node>
    <node ID="1.3" type="FN">
      <attributes />
      <edge toID="0.1" type="Terminal">
        <attributes />
        <category tag="Terminal" />
      </edge>
    </node>
    <node ID="1.4" type="FN">
      <attributes />
      <edge toID="0.2" type="Terminal">
        <attributes />
        <category tag="Terminal" />
      </edge>
    </node>
    <node ID="1.5" type="FN">
      <attributes />
      <edge toID="1.6" type="E">
        <attributes />
        <category tag="E" />
```

```
      </edge>
      <edge toID="1.7" type="C">
        <attributes />
        <category tag="C" />
      </edge>
      <edge toID="1.8" type="U">
        <attributes />
        <category tag="U" />
      </edge>
    </node>
    <node ID="1.6" type="FN">
      <attributes />
      <edge toID="0.3" type="Terminal">
        <attributes />
        <category tag="Terminal" />
      </edge>
    </node>
    <node ID="1.7" type="FN">
      <attributes />
      <edge toID="0.4" type="Terminal">
        <attributes />
        <category tag="Terminal" />
      </edge>
    </node>
    <node ID="1.8" type="PNCT">
      <attributes />
      <edge toID="0.5" type="Terminal">
        <attributes />
        <category tag="Terminal" />
      </edge>
    </node>
  </layer>
</root>
```

Such a file can be converted into a graph that looks like Figure 30. We used the UCCA package in Python to do this conversion with the following commands:

```
>>> from ucca import convert, visualization
>>> p = convert.file2passage('en-parses/1_0.xml')
>>> visualization.draw(p)
```

In the UCCA package, the content of an XML file, whether it is a sentence or a paragraph, is treated as a Passage object. In our case, it was a sentence only. The package offers tools to draw a graph from a Passage object but not directly from an XML file, which is why we needed to convert the file to a Passage first. Then, once

the picture of the graph was generated, we saved it in a PNG file. The function $visualize.draw(passage\_object)$ essentially parses information such as terminal nodes and the relations among them (represented by edges and non-terminal nodes) and collectively portray these components in a graph. We can think of the graph as an alternate way to represent the information of a parse tree, but the primary form to store this information is still an XML file whose format was previously shown.

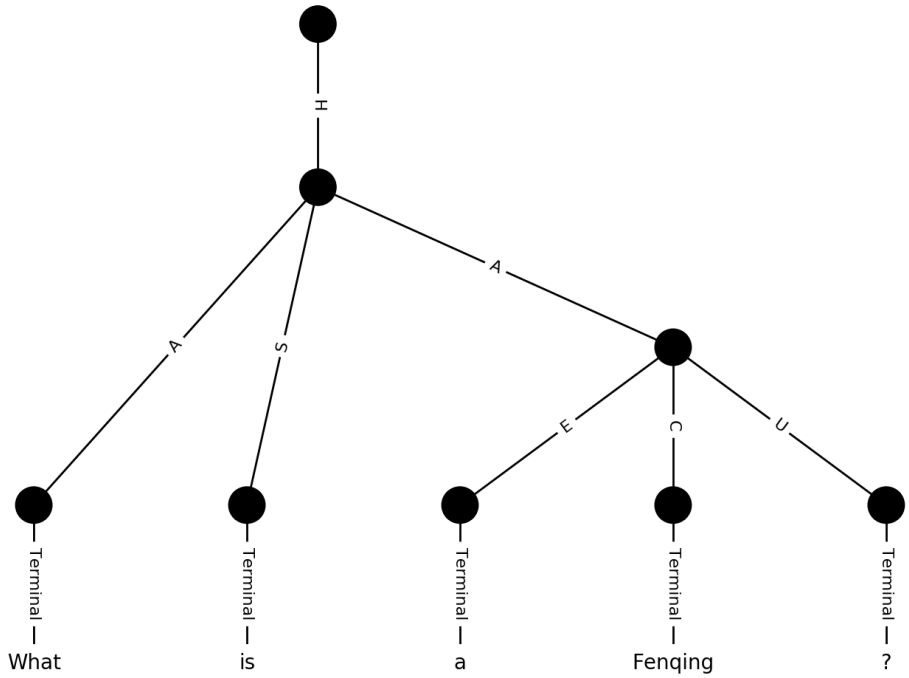

Figure 30. Graph visualized from the XML file shown above
(edge categories: *A* – Participant, *S* – State, *E* – Elaborator,
C – Center, *U* – Punctuation, more details in Table 1)

2. Converted these EN parses to VN using word alignments.

In order to do this, we wrote a script to map tokens to their one-indexed positions in each sentence. The word alignments (saved also in a mapping/dictionary) acted as a bridge between the two other mappings. This step is illustrated in Figure 31.

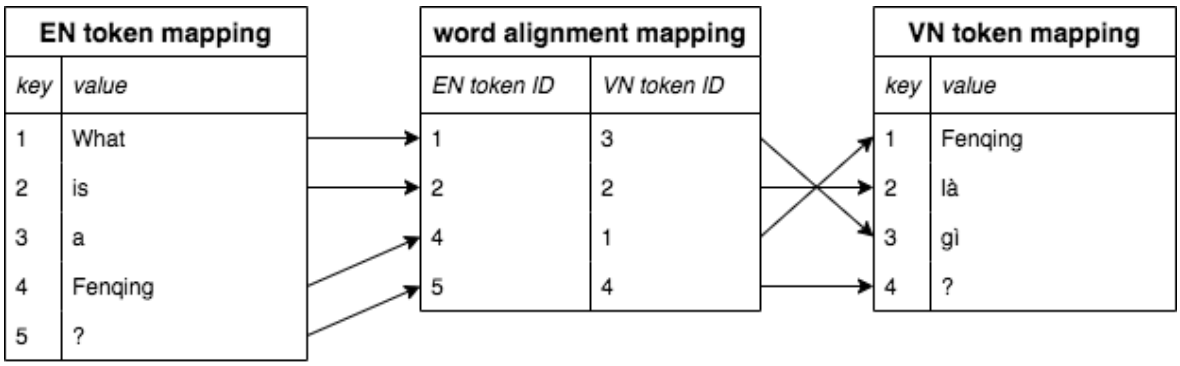

Figure 31. Mappings of EN tokens, VN tokens, and their word alignments in a bitext

Note that EN token 3 does not appear in the word alignment mapping because it does not have a VN counterpart. We refer to these tokens as unmatched tokens, which would be handled in the next step.

This conversion task, however, was not as simple as that because we had to account for one-to-many relationships (i.e., one EN token matched to several VN tokens), many-to-one relationships (i.e., several EN tokens matched to one VN token), and many-to-many relationships (i.e., several EN tokens matched to several VN tokens). To handle these non-one-to-one relationships, we created another dictionary keyed by each EN token, since it was from each EN token that we wanted to convert to a VN token or phrase. If one-to-many relationship, the value would be a list of VN tokens. If many-to-many, each EN token in that phrase would be mapped to the same list of VN tokens. And if many-to-one, each EN token in the phrase would be mapped to the same VN token.

To give an example, let us consider the following bitext:

(33) Is the United States a power in decline?

(34) Có phải thế lực của Hoa Kỳ đang suy yếu?

Its word alignments are [1-1,2], [3,4-6,7], [6-3,4], [8-9,10], which include several one-to-many relationships (e.g., 1 is mapped to 1, 2) and a many-to-many relationship (i.e., 3, 4 are mapped to 6, 7), so the dictionary will look like Figure 32:

word alignment mapping	
EN token ID	*VN token IDs*
1	1, 2
3	6, 7
4	6, 7
6	3, 4
8	9, 10

Figure 32. Example word alignment mapping

of one-to-many and many-to-many relationships

Note that for many-to-many and many-to-one relationships, duplication would happen when all the EN tokens were replaced by their VN counterparts, as we can see with value (6, 7) appearing twice in the dictionary in Figure 32. This issue would be addressed in a later step.

3. Handled unmatched EN tokens.

There were EN tokens that were not matched to any VN tokens, and vice versa. In Figure 32, for example, EN tokens 2, 5, and 7 are unmatched as well as VN tokens 5 and 8. In the converted graph, these EN tokens therefore do not have any corresponding VN tokens to replace them, so the graph looks like Figure 33.

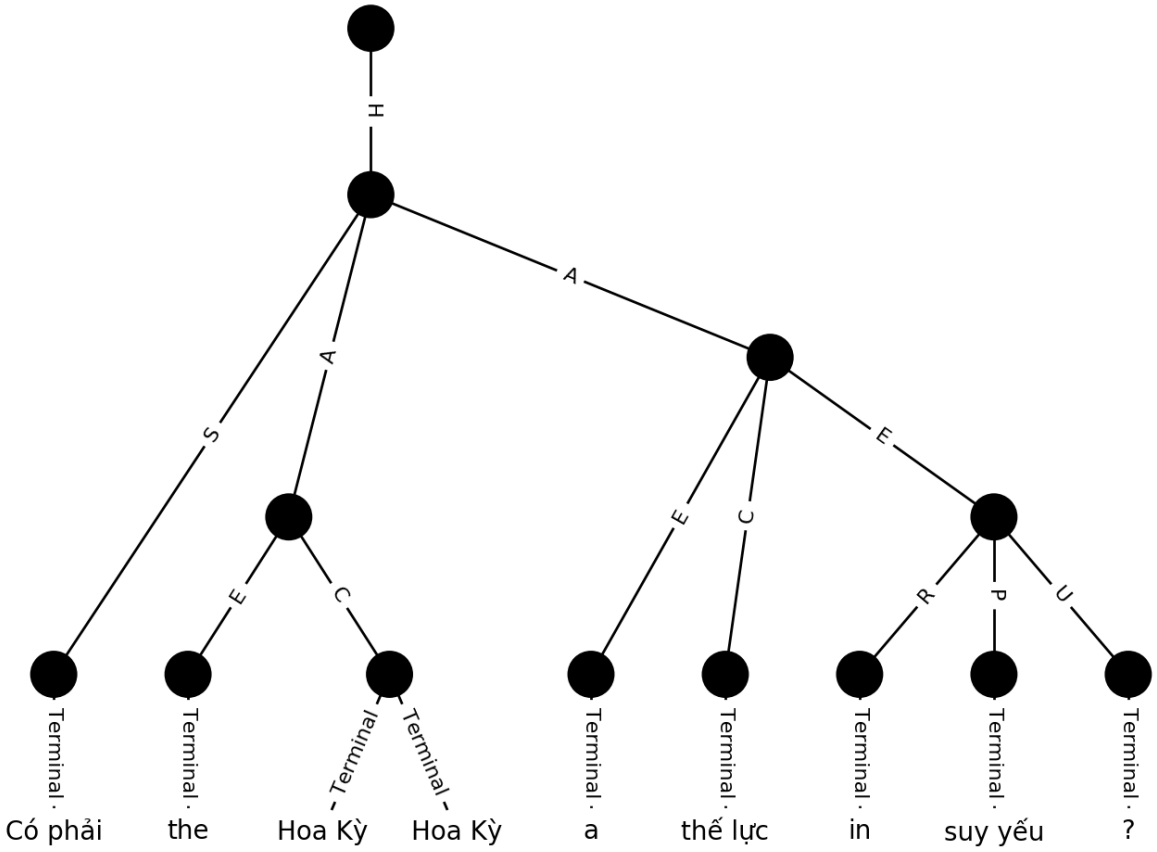

Figure 33. Converted graph for bitext (33)-(34)

Notice the terminal nodes “the”, “a”, and “in” corresponding to EN tokens 2, 5, and 7 were not replaced by any VN tokens. A script was written to remove these nodes.

The removal of unmatched nodes required removing also ancestor nodes that only led to each of these nodes and no others, thus was implemented by adapting the breadth-first search (BFS) algorithm to traverse up the tree (instead of downwards) to search for such ancestors. As for the reverse situation (i.e., unmatched VN tokens), there was nothing to be done since we could not automatically insert VN tokens that had no semantic annotation basis (which could only be done manually), so this is one of the losses in accuracy that we trade for auto-generation of annotated resources using a cross-lingual approach. Therefore, we do not see unmatched VN tokens at positions 5 and 8 (i.e., “của” and “đang”) appear in the graph in Figure 33. The VN parse after unmatched nodes (and their single-child ancestors, if any) were removed is shown in Figure 34.

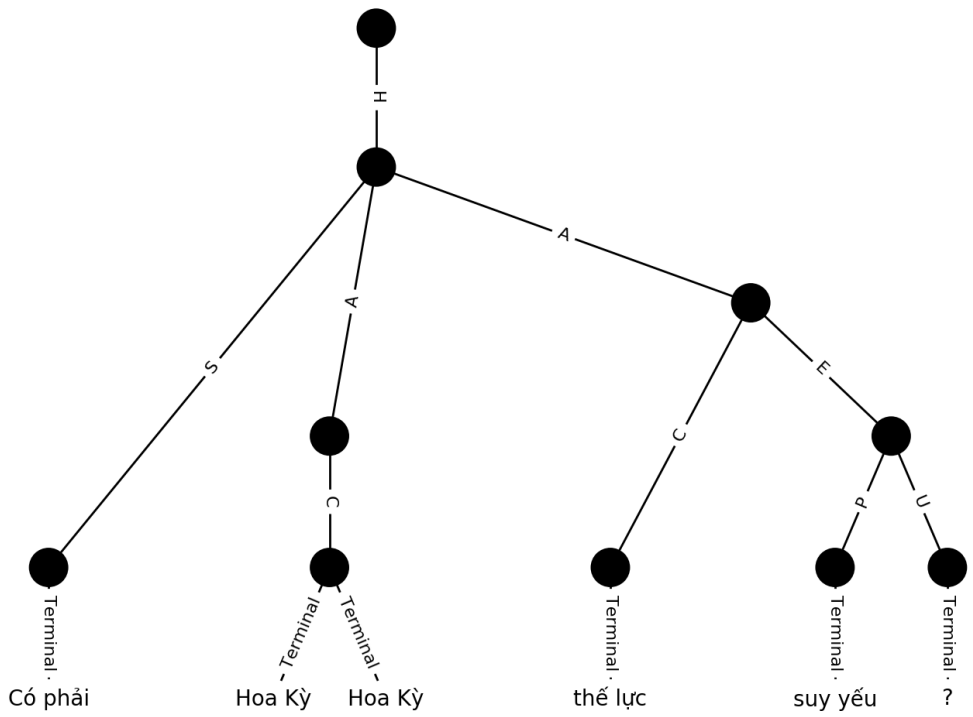

Figure 34. Converted graph for bitext (33)-(34)
after unmatched nodes were trimmed

4. Removed duplicates.

Our preprocessing also generated duplicates in the converted UCCA parses for Vietnamese sentences, as seen in Figure 33 and Figure 34 with terminal node "Hoa Kỳ" (corresponding to VN token IDs 6 and 7) being repeated twice. Therefore, we wrote a script to remove them through the following process.

First, we identified duplicates, which were keys that had the same value in the dictionary created in step 2, and saved them into a new dictionary keyed by the shared VN tokens. Each value would then be a list of EN tokens that shared a certain set of VN tokens (i.e., their key). For example, the new dictionary created based on Figure 32 will contain a single key-value pair: (6, 7) → [3, 4], since the only duplicated value in Figure 32 is (6, 7) which is mapped from keys 3 and 4.

After that, we established different cases and criteria for which nodes to remove in each case:

a) If all duplicate tokens shared a parent node and no siblings other than themselves, we would keep any one node, remove the rest, connect the selected node directly to its grandparent (since now this node acts exactly as its parent node), and delete the parent.
b) If all duplicate tokens shared a parent node but had other non-duplicate siblings, then we would keep one that had higher precedence (e.g., Center (*C*) > Elaborator (*E*)). This precedence was determined based on the category table of UCCA's foundational layer (Table 1).
c) If the duplicates did not share a parent node and had different numbers of parents, we would choose one that had the greatest number of parents (i.e., more relations were dependent on it).
d) If the duplicates did not share a parent node but had the same number of parents (we automatically assumed each had a single parent), then we would choose one whose parent's ID was smaller, which indicated that the parent was higher in the tree.

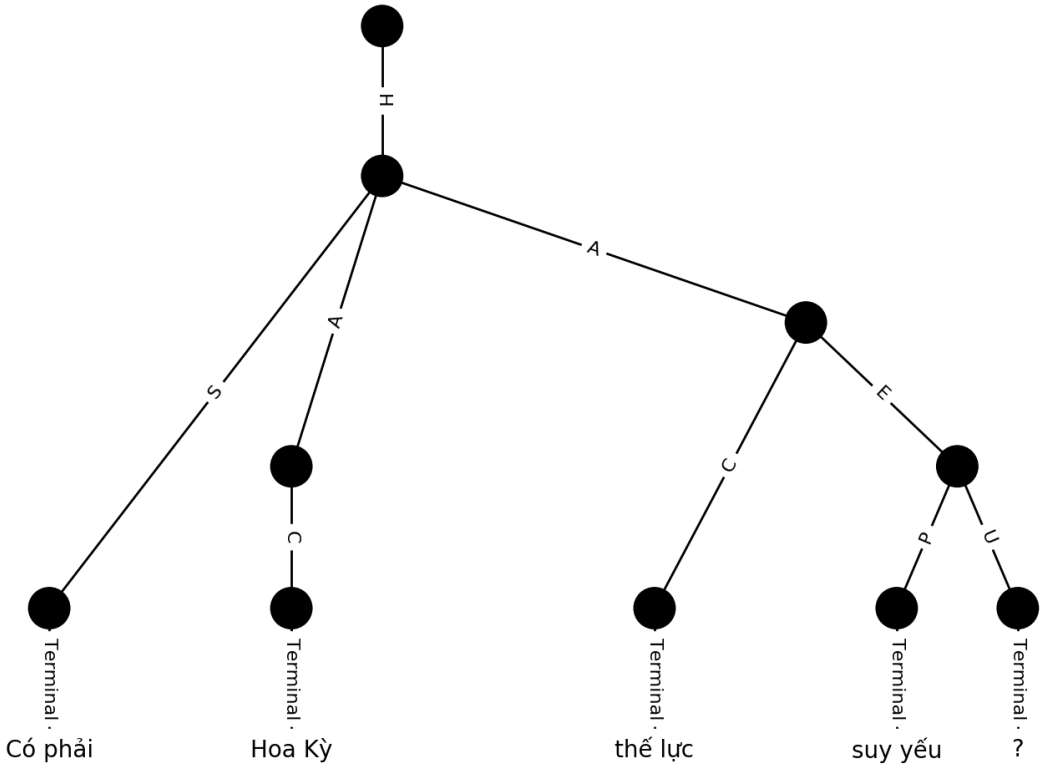

Figure 35. Parse tree for VN sentence (B) after duplicates are removed

5. Relabeled node IDs to be sequential.

As a result of node removals, the remaining node IDs were no longer sequential. Although not affecting the structure of a parse tree per se, this caused problems when the model was validated on the development (dev) set. To be more specific, the model attempts to generate a semantic parse for each sentence in the dev set. This parse is in turn compared against the parse that already exists for that sentence (which is supposedly the "gold standard" in this context but indeed is only a product of our best attempt at cross-lingual conversion using word alignments and is probably not the most accurate). However, the generated parse has sequential node IDs while the existing one does not, for which the model will complain that there is a mismatch in node IDs and halt its validation process.

To remedy this issue, we relabeled the ID of each terminal node according to its index in the (one-indexed) list of terminal nodes. IDs of non-terminal nodes remained the same since their matching was not checked and changing them would also require a significant amount of alteration to our current parses.

6.3. Model Training

After duplicate nodes had been removed and all parse trees trimmed through the processes described in 6.2, we ended up with a total of 38,160 labeled sentences. The number had decreased since certain sentences had to be skipped due to parsing complications. In fact, such removals were to ensure that the examples we finally had for training were exactly what our cross-lingual preprocessing had been set up to obtain. This amount of quality data was sufficient to train a decent model.

Next, the data (i.e., 38,160 XML files containing converted VN parses) was split into training (train), development (dev), and test sets. The test set consists of 10 hand-picked sentences since we wanted to select those with reasonable numbers of words (i.e., not too long or too short) and contain language expressions different from those in their English counterparts. The train set contains 90% of the remaining 38,150 examples and the dev set 10%, picked randomly using *train_test_split* from Python's *sklearn* library. Relevant statistics are reported in Table 4.

	Train	**Dev**	**Test**
# Sentences	34,335	3,815	10
# Actually Used	34,328	3,815	10

Table 4. Statistics on train, dev, test sets

As mentioned in 5.3, TUPA offers three choices of model types to train, so we chose BiLSTM for our model since it outperforms the other two in most cases and was also used for the pretrained model (Hershcovich et al., 2017). The TUPA package provides a command-line interface to train the parser, for which we only needed to specify as input the relative paths to train and dev folders as well as the model type (i.e., BiLSTM). Then, a model with the BiLSTM architecture was created and then trained and validated for 50 epochs, each taking 15870s (≈ 4 hours 24 minutes) to run on average. During training in each epoch, the model parameters were adjusted based on incorrect predictions it had made in the last epoch. After being trained on 34,328 valid passages, its performance was measured on 3,815 passages in the dev set using the F1 score, which we will explain in the next subsection. If the F1 score of the current epoch was higher than the best score achieved so far, the model files (which hold the construction and parameter settings of the best model) would be updated with the new parameters adjusted in this current epoch. Figure 36 shows this whole process in action for epoch 49.

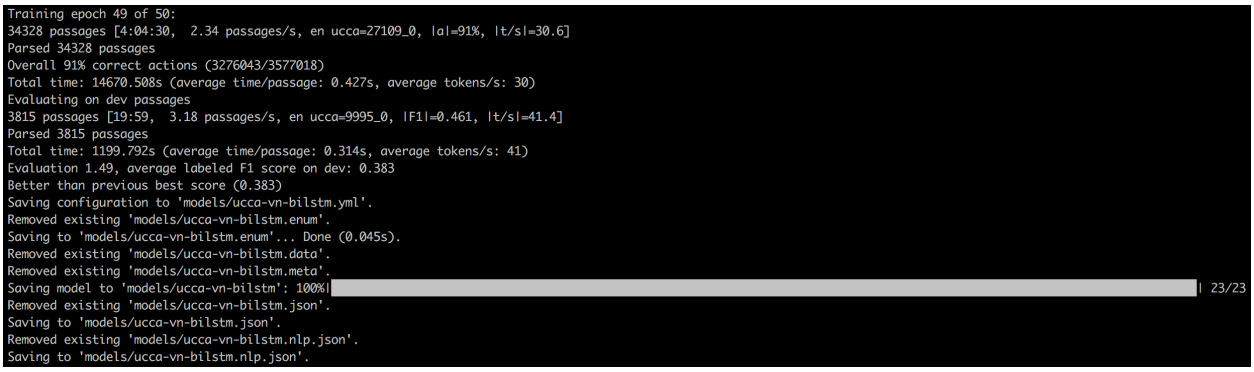

Figure 36. Model was trained and validated in an epoch
(and saved if it obtained a higher F1 score than current best score)

6.4. Evaluation

As we can see in Figure 36, TUPA evaluated the model's performance automatically after each validation step using the F1 score, a popular measure of a test's accuracy for a classification task (i.e., a task to predict the class for an unlabeled example). In this subsection, we will explain how exactly it is calculated in the UCCA graph setting. The score is essentially the *harmonic mean* of *precision* and *recall* and ranges from 0 to 1, with 0 reflecting the worst performance and 1 the best (i.e., perfect precision and recall). Now, we will define the italicized terms above and arrive at the formula for F1.

- In a binary classification task that only has two classes to predict (usually positive and negative), the performance of a classification model (i.e., a classifier) can be summarized in a *confusion matrix*, which is a table that whose format looks like Figure 37:

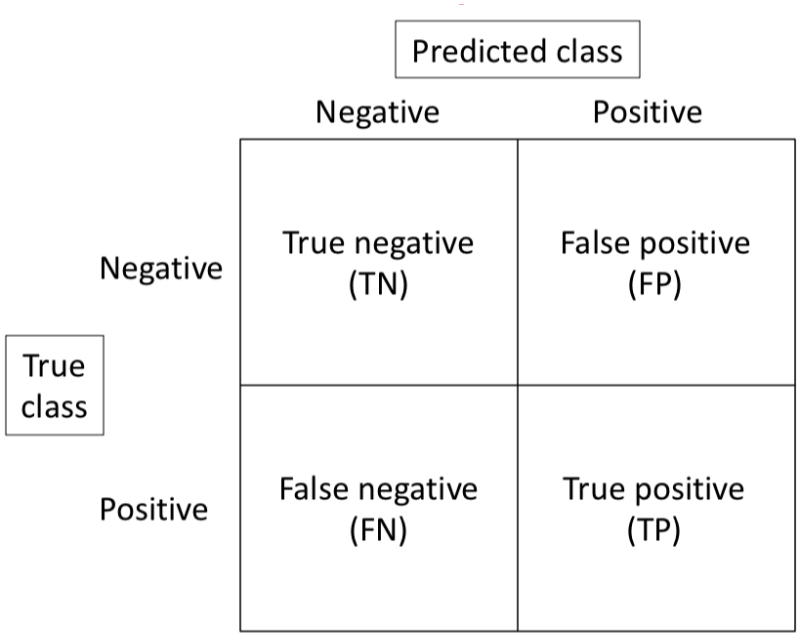

Figure 37. Format of a confusion matrix (Mathieson, 2019)

In an actual confusion matrix, the TN, FN, FP, and TP cells would contain numbers of results that respectively are true negative (i.e., true label is negative and prediction is also negative), false negative (i.e., true label is positive but prediction is negative), false positive (i.e., true label is negative

but prediction is positive), and true positive (i.e., both true label and prediction are positive).

- Precision is the number of correct positive results (i.e., true positive) divided by the number of all results classified as positive by the model (which is the second column in a confusion matrix, i.e., false positive + true positive). Thus, the formula for precision is:

$$precision = \frac{true\ positive}{false\ positive + true\ positive}$$

as also demonstrated in Figure 38.

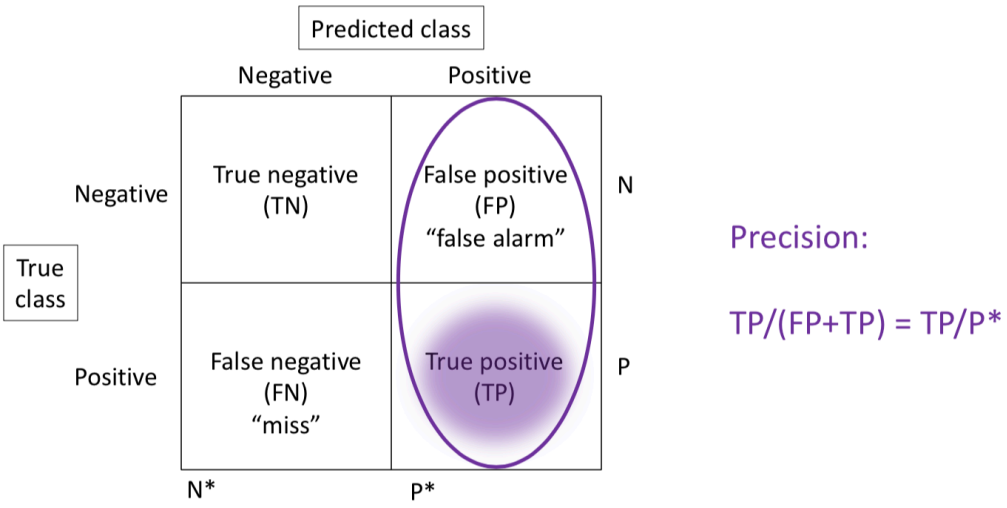

Figure 38. Precision in a confusion matrix (Mathieson, 2019)

The idea behind precision is to measure the percentage of results that are relevant (i.e., the percentage of the returned results that are actually what we are looking for).

- Recall is the number of correct positive results (i.e., true positive) divided by the number of all samples whose true labels are positive (which is the second row in a confusion matrix, i.e., false negative + true positive). It is calculated with the following formula:

$$recall = \frac{true\ positive}{false\ negative + true\ positive}$$

and also demonstrated in Figure 39.

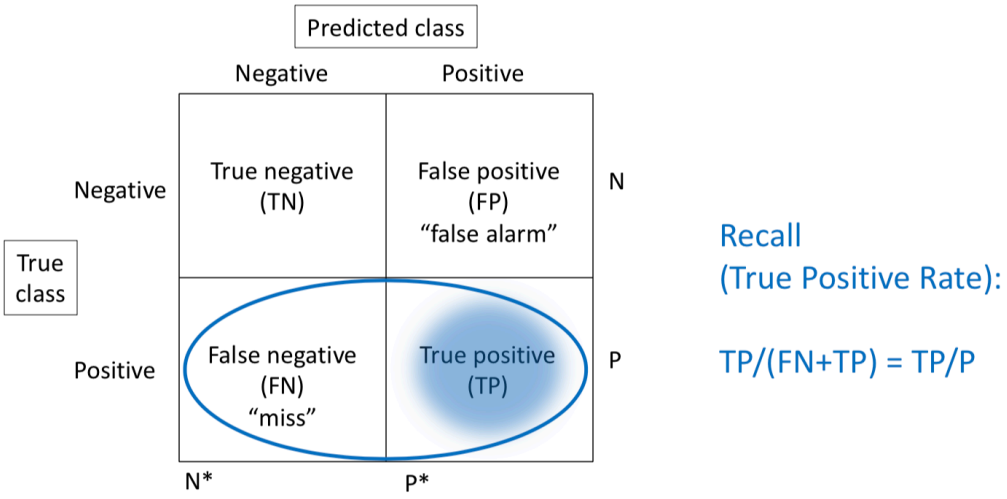

Figure 39. Recall in a confusion matrix (Mathieson, 2019)

The idea behind recall is to measure the percentage of relevant results that are correctly classified (i.e., the percentage of the results we are looking for that are actually returned).

- The harmonic mean H of positive real numbers $x_1, x_2, \dots, x_n$ is calculated as:

$$H = \frac{n}{x_1^{-1} + x_2^{-1} + \cdots + x_n^{-1}}$$

- Therefore, the formula for F1 score is:

$$F1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

However, the classification task of our model was not binary, so precision and recall needed to be defined in a different way that would still rely on the idea behind each of them. In order to do so, Hershcovich et al. (2017) first devised a measure to compare UCCA structures based on *mutual edges* (i.e., edges that two or more UCCA graphs share) as follows:

- Let $G = (V, E, \ell)$ be a UCCA graph where V denotes the set of nodes, E denotes the set of edges, and $\ell : E \rightarrow L$ is the label function with L representing the set of possible labels.

- From the definition above, $G_p = (V_p, E_p, \ell_p)$ and $G_g = (V_g, E_g, \ell_g)$ represent the predicted and gold-standard graphs respectively over the same sequence of terminals $W = \{w_1, w_2, \dots, w_n\}$.
- In either graph, an edge $e = (u, v)$ with u being the parent node and v the child has its yield $y(e) \subseteq W$ defined as a subset of terminals in W that are descendants of v.
- Then, the set of *mutual edges* between G_p and G_g is defined as:

$$M(G_p, G_g) = \{(e_1, e_2) \in E_p \times E_g \mid y(e_1) = y(e_2) \wedge \ell_p(e_1) = \ell_g(e_2)\}$$

  which translates as a set of tuples of edges (e_1, e_2) where e_1 is an edge in the predicted graph G_p and e_2 is an edge in the gold standard G_g such that e_1 and e_2 have the same yield and the same label function, i.e., e_1 and e_2 are *mutual*.

In our graph problem, mutual edges (i.e., $M(G_p, G_g)$) are indeed similar to true positives in a binary classification setting, while edges in the predicted graph G_p (i.e., E_p) are essentially results predicted positive (i.e., false positive + true positive) and edges in the gold-standard graph G_g (i.e., E_g) are results whose correct labels are positive (i.e., false negative + true positive). Thus, precision and recall were calculated by dividing $|M(G_p, G_g)|$ by $|E_p|$ and $|E_g|$ respectively, and our F1 score was their harmonic mean (Hershcovich et al., 2017).

7. Results and Discussion

This section reports on the results of our final model and discusses potential reasons for its demonstrated performance.

7.1. Results

The model has a 91% *training* accuracy. The F1 score measured on our dev set started from 0.249 after epoch 1 and improved to the highest score of 0.383 at epoch 49. After 50 epochs, the model with the best parameters (i.e., one that gave the highest F1 score on the dev set) was saved, so we can think of this whole process as the *parameter tuning* step.

Once we had the best set of parameters for our model, we trained it one more time on both the train and dev sets (i.e., 38,143 examples in total), which is the official training step. In fact, these two datasets together constitute all training data, which the model was allowed to use to fit and tune its parameters. Note that, thus far, our model had never seen the test set before. It was only at this point that it could finally be evaluated on the 10 held-out sentences. The F1 score on this data is 0.481, indicating a 48.1% *test* accuracy achieved by our final model.

7.2. Discussion

This accuracy is as expected, if not better. Having identified several potential losses in accuracy, we anticipated it to be around 20% at the lowest. First of all, the generated EN parses themselves were only around 70% correct since the accuracy of TUPA's pretrained model is 73.5%. Transferring these parses to Vietnamese mainly using word alignments and without any manual corrections afterwards guaranteed a further reduction in accuracy. As mentioned before, this is what we have to compromise when using a cross-lingual approach to auto-generate annotated resources, which is the goal of this experimental research. Moreover, the converted VN parses (i.e., our training data) were not quite in their best form either, which we will discuss in the next section regarding potential improvements. For these reasons, the accuracy of our model was projected to fall within the range of 20-50%, so a 48.1% score is well towards the higher end.

Nevertheless, it is worth noting that this accuracy is only relative to the converted VN parses that were auto-generated through our cross-lingual process, which are not entirely accurate themselves. In order to be confident that the accuracy score closely reflects how the model's results compare to true gold standards, we have to ensure the quality of the converted VN parses is as close to that of gold-standard parses as possible.

As we further examined the results by comparing the predicted graphs for the 10 test sentences to their "gold standard" graphs (i.e., their converted parses), we noticed another problem that is less technical but rather language-specific. When the

model generates a parse for a sentence, it treats every token as a word (since its tokenization process is simply done by splitting the string by space) and attempts to assign a category to it, as we can see in Figure 40 for a test sentence.

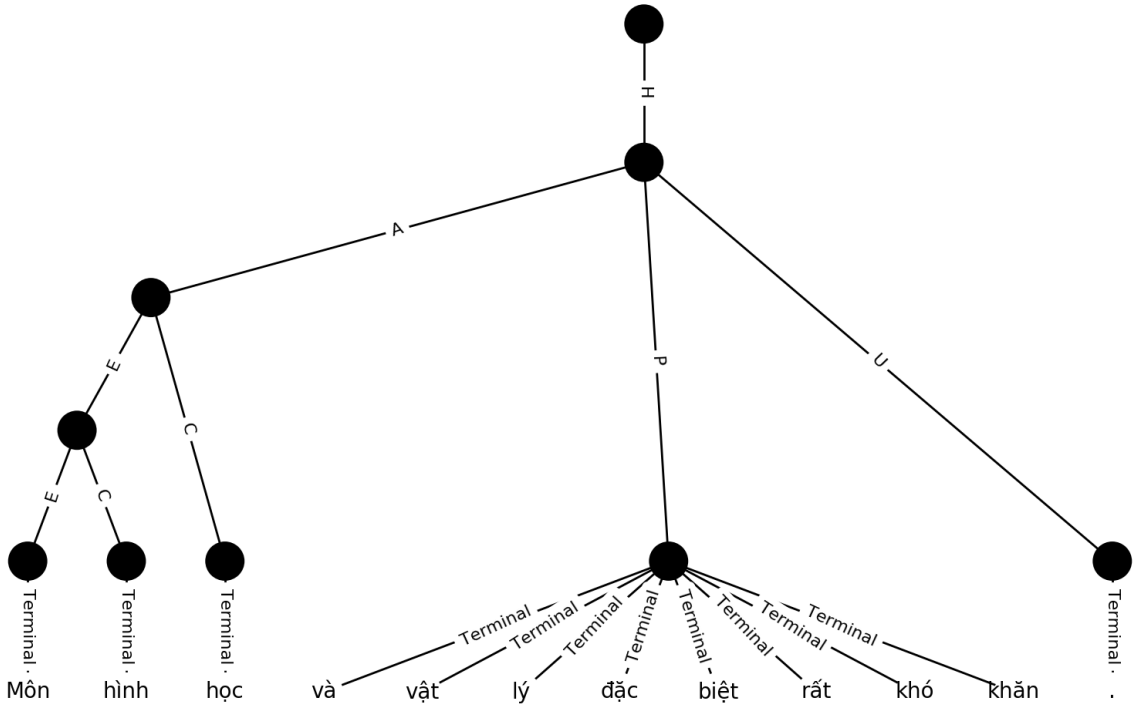

Figure 40. Predicted graph for a test sentence

However, in Vietnamese, a word can contain multiple space-separated tokens (i.e., a multi-word), which our conversion process handled by combining into one terminal node (e.g., “môn hình học” or “vật lý” in Figure 41).

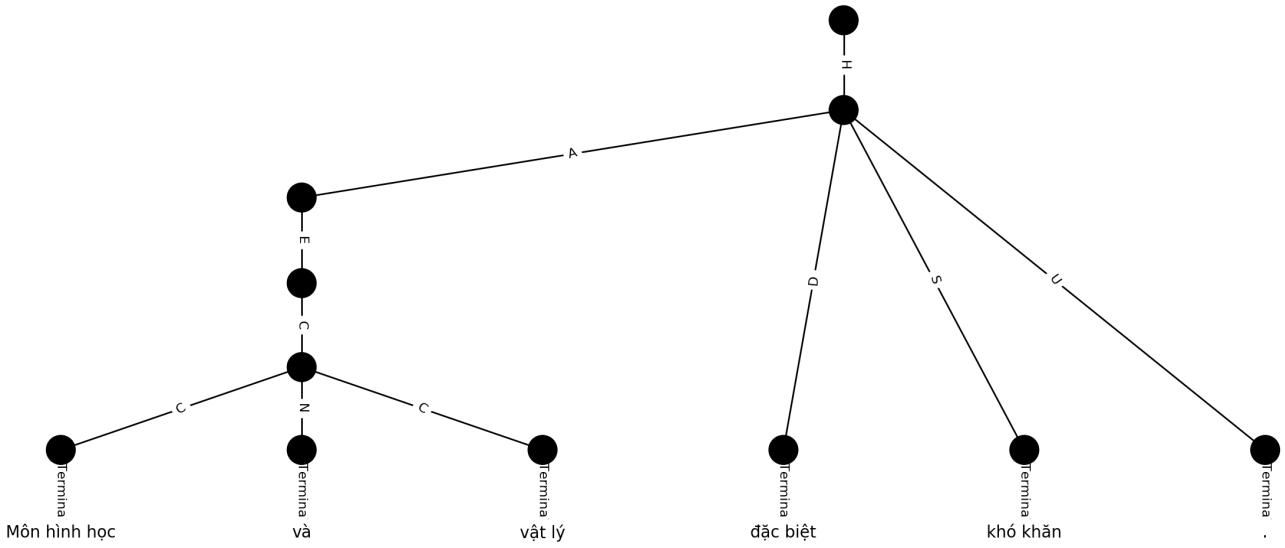

Figure 41. “Gold-standard” graph (i.e., converted graph)
for the same test sentence

As our model was trained on such converted parses, multi-words were saved as part of the model’s features but would not actually be useful because input sentences would only have single-token words as features due to how the model tokenizes strings. Fortunately, we can use chunking to overcome this problem, which will be discussed in the section that follows.

8. Future Work

Our analysis has revealed that the biggest area for improvement is the training set. For future work, one can attempt reordering the words in each converted VN parse to preserve the original order of the VN sentence and represent its semantics more accurately. Doing so should improve the performance of the model as well, since it can learn certain syntactic phenomena that frequently occur in Vietnamese (e.g., nouns preceding adjectives, topic-comment structure, wh-in-situ, etc.) and apply them to better parse unlabeled sentences. As we can see through the reported statistics, a few thousand sentences were excluded from the final training data because we encountered complications while preprocessing them. Taking the time to fine-tune our preprocessing step so that it handles these errors can also help enlarge the training set.

When discussing the problem of multi-words vs. single-token words in 7.2, we also suggested chunking which is a process to extract phrases from a text. Particularly for Vietnamese, it is helpful in identifying multi-words right from the beginning so that one can decide how to preprocess them. A tool that has already been developed to accomplish this task is VietChunker (VLSP Projects, 2020), which handles multi-words by replacing spaces in-between with underscores (e.g., "môn hình học" would become "môn_hình_học"). Whichever chunking method one chooses, it should be applied to both train and test sentences before they are used to fit or evaluate the model.

One can take a step further to look through the converted VN parses and document the most common, if not all, errors that we missed in preprocessing the nodes and edges. For example, in Figure 41, the two non-terminal nodes above node "và" and their incident edges marked with *E* and *C* are completely unnecessary and should have been removed, but our conversion process did not identify and handle cases like this. Since 38,160 is a large number of parses to go through, one can choose to look at shorter sentences whose relations are not as complicated instead and make sure their parses are as correct as they can be.

A few minor improvements would have to do with identifying and correcting errors in our source data, EVBNews, since it is not a perfect corpus. For example, we came across some word alignments in which token IDs were not correctly numbered, which led to incorrect mappings of EN-VN tokens. More fundamental improvements such as faster training time or better predictions of node categories (to produce higher-accuracy EN parses in the first place) would lie in the construction of TUPA, which is beyond the scope of this thesis. However, the main developer of this parser, Daniel Hershcovich, who has provided helpful guidance for the model training process in our research, is a great point of contact if one is interested in investigating further the architecture behind TUPA.

9. Conclusion

This research has provided background on the structures of three MR schemes – UDS, AMR, and UCCA, assessed the potential of each MR in a cross-lingual setting by analyzing its linguistic strengths and weaknesses when applied to a non-English (particularly, Vietnamese) text, designed a cross-lingual process to generate meaning parses for Vietnamese sentences from those of their English counterparts, produced a semantically annotated dataset for the language with over 38K sentences, and trained a model provided by TUPA using this dataset. The final model has an F1 score of 48.1% and is the first semantic parser, albeit experimental, to have been developed for Vietnamese. For future work, we have identified both linguistic and computational areas that can be improved to fine-tune our cross-lingual process and arrive at a higher-confidence accuracy for the parser. Overall, this thesis lays the groundwork for building a semantic parsing model for any given language using cross-lingual processes, which has promising potential in auto-generating annotated corpora for low-resource languages.

Bibliography

Allen B. Tucker. 2002. Semantics – meaning representation in NLP. *Notes on Natural Language Processing (NLP).* [Online]. Available: http://www.bowdoin.edu/~allen/nlp/nlp6.html.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*. pages 2442–2452. http://aclweb.org/anthology/P16-1231.

Daniel Hershcovich, Omri Abend, and Ari Rappoport, (2017). "A Transition-Based Directed Acyclic Graph Parser for UCCA," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. SemEval 2019 Task 1: Cross-lingual semantic parsing with UCCA. arXiv:1903.02953, 2019.

Daniel Jurafsky and James H. Martin, *Speech and language processing*. Harlow: Pearson, 2009.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*. pages 740–750. http://aclweb.org/anthology/D14-1082.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488, 2015.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. *Proceedings of S2MT*, 11–22.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency using bidirectional LSTM feature representations. *TACL* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. Comput. Linguist. 29, 1 (March 2003), 19–51. DOI:https://doi.org/10.1162/089120103321337421.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Linh Ha and Huyen Nguyen. 2019. A case study on meaning representation for Vietnamese. *Proceedings of the First International Workshop on Designing Meaning Representations*, 2019.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. *Proceedings of LREC*, 4585–4592, 2014.

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

P. T. Nguyen, L. V. Xuan, T. M. H. Nguyen, V. H. Nguyen, and P. Le-Hong. Building a large syntactically-annotated corpus of Vietnamese. In *Proc. of the 3rd Linguistic Annotation Workshop, ACL-IJCNLP*. pages 182–185.

Quoc Hung Ngo, Werner Winiwarter, and Bartholomaus Wloka. 2013. EVBCorpus – A multi-layer English-Vietnamese bilingual corpus for studying tasks in comparative linguistics. In Proceedings of the 11th Workshop on Asian

Language Resources (11th ALR within the IJCNLP2013), pp. 1-9. Asian Federation of Natural Language Processing.

Quoc-Hung Ngo, Werner Winiwarter. 2012. Building an English-Vietnamese bilingual corpus for machine Translation, International Conference on Asian Language Processing 2012 (IALP 2012), pp. 157-160. IEEE Computer Society.

Robert M. W. Dixon. *The rise and fall of languages*. Cambridge: Cambridge University Press, 2006.

Sara Mathieson. (2019) *CS 360: Machine Learning, Lecture 13*. http://cs.haverford.edu/faculty/smathieson/teaching/f19/lecs/lec13/lec13.pdf.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual semantic parsing. arXiv:1804.08037, 2018.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. *Proceedings of EMNLP*, 1664–1675.

UCCA website. 2020. *Universal Conceptual Cognitive Annotation (UCCA)*. [Online]. Available: https://universalconceptualcognitiveannotation.github.io/.

UDS website. 2020. *Universal Dependencies*. [Online]. Available: https://universaldependencies.org/.

VLSP Projects. 2020. *Vietnamese Language Resources, VLSP*. [Online]. Available: https://vlsp.hpda.vn/demo/?page=resources&lang=en.

Yonggang Deng. (2005). Bitext Alignment for Statistical Machine Translation. PhD thesis, Johns Hopkins University.