# Knowledge Authoring with Factual English*

Yuheng Wang Giorgian Borca-Tasciuc Nikhil Goel
Paul Fodor Michael Kifer
Department of Computer Science, Stony Brook University
Stony Brook, NY, USA
{yuhewang,gborcatasciu,nigoel,pfodor,kifer}@cs.stonybrook.edu

Knowledge representation and reasoning (KRR) systems represent knowledge as collections of facts and rules. Like databases, KRR systems contain information about domains of human activities like industrial enterprises, science, and business. KRRs can represent complex concepts and relations, and they can query and manipulate information in sophisticated ways. Unfortunately, the KRR technology has been hindered by the fact that specifying the requisite knowledge requires skills that most domain experts do not have, and professional knowledge engineers are hard to find. One solution could be to extract knowledge from English text, and a number of works have attempted to do so (OpenSesame, Google's Sling, etc.). Unfortunately, at present, extraction of logical facts from unrestricted natural language is still too inaccurate to be used for reasoning, while restricting the grammar of the language (so-called controlled natural language, or CNL) is hard for the users to learn and use. Nevertheless, some recent CNL-based approaches, such as the Knowledge Authoring Logic Machine (KALM), have shown to have very high accuracy compared to others, and a natural question is to what extent the CNL restrictions can be lifted. In this paper, we address this issue by transplanting the KALM framework to a neural natural language parser, mSTANZA. Here we limit our attention to authoring facts and queries and therefore our focus is what we call *factual* English statements. Authoring other types of knowledge, such as rules, will be considered in our followup work. As it turns out, neural network based parsers have problems of their own and the mistakes they make range from part-of-speech tagging to lemmatization to dependency errors. We present a number of techniques for combating these problems and test the new system, KALM$^{\text{FL}}$ (i.e., KALM for factual language), on a number of benchmarks, which show KALM$^{\text{FL}}$ achieves correctness in excess of 95%.

## 1 Introduction

Much of the human knowledge can be captured in knowledge representation and reasoning (KRR) systems that are based on logical facts and rules. Unfortunately, translating human knowledge into the logic form that can be used by KRR systems requires well-trained domain experts who are hard to come by.

One popular idea is to use natural language (NL) to represent knowledge, but current technology (e.g. OpenSesame [13], SLING [12]) for converting such statements into logic has rather low accuracy. A possible fix to this problem is to author knowledge via sentences in controlled natural languages (CNLs), such as ACE used in Attempto [6]. These CNLs are fairly rich and algorithms exist for converting CNL sentences into logic facts. Unfortunately, CNLs are also very restrictive, hard to extend, and require significant training to use. Furthermore, both CNLs and the more general NLP systems cannot recognize sentences with identical meaning but different syntactic forms. For example, "*Mary buys a car*" and "*Mary makes a purchase of a car*" would be translated into totally different logical representations by

*Research partially funded by the NSF Grant 1814457. We would also like to thank Nathanael Payen for his contribution to software development for this work.

Y. Lierler, J.F. Morales, C. Dodaro, V. Dahl, M. Gebser, T. Tekle (Eds.):
International Conference on Logic Programming 2022 (ICLP'22).
EPTCS ??, 2022, pp. 107–122, doi:10.4204/EPTCS.??.10

most systems, which renders logical inference mechanisms unreliable at best. This problem is known as *semantic mismatch* [7].

Recently, the Knowledge Authoring Logic Machine (KALM) [8] was introduced to solve the above semantic mismatch problem, but KALM was based on Attempto's ACE and therefore inherited all the aforesaid problems with CNLs. In this paper, we address the problems associated with controlled languages by transplanting the KALM framework to a neural NL parser, mSTANZA, which is a modified STANZA [11] version with multiple, ranked outputs. Of course, to turn English into an authoring tool for KR one still needs to impose some restrictions on the language. For instance, "Go fetch more water" is a command that does not convey any factual information that can be recorded in a knowledge base (except, perhaps, those based on rather esoteric logics). In this paper, we focus on English sentences suitable for expressing facts and queries and correspondingly identify a class of English sentences, which we call *factual*. These sentences can be translated into logic and the aforesaid semantic mismatch problem is solved for such sentences. Unlike CNLs, factual sentences need little training as long as the author keeps focus on knowledge representation rather than fine letters.[1]

To increase the accuracy, we had to mitigate a slew of issues that are common to neural parsers, and we describe our solutions. These include the mistakes in part-of-speech and dependency parsing. The new system, KALM$^{\text{FL}}$ (KALM for factual language)[2], is tested on a number of benchmarks, which show that KALM$^{\text{FL}}$ for factual English achieves correctness in excess of 95%, very close to the original KALM for the Attempto's CNL.

The paper is organized as follows: Section 2 reviews the KALM framework, Section 3 defines factual sentences, Section 4 proposes mSTANZA and the new KALM$^{\text{FL}}$ framework, Section 5 shows the evaluation settings and results, Section 7 concludes the paper.

## 2 The KALM Framework

KALM [7] is a semantic framework for scalable knowledge authoring. KALM users author knowledge using CNL sentences (Attempto's ACE, to be specific) and KALM ensures that semantically equivalent sentences have identical logical representations through the use of the frame semantics [4]. The framework is depicted in Fig. 1.

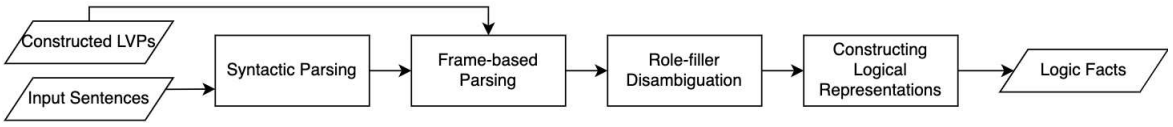



Figure 1: The KALM framework

**Syntactic Parsing.** KALM uses Attempto Parsing Engine (APE) to extract the syntactical information from sentences, including the part-of-speech (POS) for each word and the grammatical dependency relations between pairs of words. All extracted information is represented by a set of logical terms known as Discourse Representation Structure (DRS) [5]. Here is an example of a DRS.

**Example 1** DRS for the sentence "*Mary buys a car*".

```
object(A,mary,uncountable,na,eq,1)-1/1.    predicate(C,buy,A,B)-1/2.
object(B,car,countable,na,eq,1)-1/4.
```

[1] Our followup work will consider more general sentences, such as those suitable for expressing rules.

[2] https://github.com/yuhengwang1/kalm-fl

where `A` and `B` are identifiers for the *Mary*- and *car*-entities, respectively, and `C` is the *buy*-event. The DRS also relies on the predicates `object/6` and `predicate/4` (in `p/N`, `N` denotes the number of arguments in predicate `p`). An `object`-fact represents an entity—a noun-word with some properties (e.g., countable or uncountable, quantity). A `predicate`-fact represents an event—a verb-word and its participating entities. Additional predicates, `property/3`, `relation/3`, `modifier_adv/3`, `modifier_pp/3`, and `has_part/2`, are also used in DRS for representing other syntactic relations. An expression like `1/4` in a DRS `object`-fact indicates the sentence Id (i.e., 1) and the token Id (here 4) described by the object fact in question.

**Logical Valence Pattern.** Frame-based parsing requires logical valence patterns (LVPs), which are constructed from *training* sentences annotated by knowledge engineers. Inspired by FrameNet [1], the semantics of each training sentence is represented by a *frame*. Each frame defines one or more related semantic relationships among entities, where each entity plays a particular *role*. A frame can be triggered by its associated "triggering words," called *lexical units (LU)*. Below is an example of a training sentence annotated with frame semantics.

**Example 2** The training sentence "*Mary buys a car*".

```
train('Mary buys a car','Commerce_buy','LUIndex'=2,
      ['Buyer'=1+required,'Goods'=4+required],[[purchase,verb],[acquire,verb]]).
```

The above says several things. (i) "*Mary buys a car*" is represented by the `Commerce_buy` frame, (ii) the LU is the 2nd word, `buy.verb`, (iii) the 1st word *Mary* fills the role `Buyer` and the 4th word *car* fills the role `Goods`, and (iv) the words `purchase.verb` and `acquire.verb` can also trigger this frame. Combining this and the DRS in Example 1, KALM learns that starting from the LU `buy.verb` (i.e., `predicate(C,buy,A,B)`), the `Buyer` *Mary* (i.e., `object(A,mary,uncountable,na,eq,1)`) can be found by locating the subject (i.e., the 3rd argument in `predicate/4`) of `buy.verb`. As a result, the pattern `verb->subject` for finding the role-filler for `Buyer` is learned. Similarly, the pattern `verb->object` to find `Goods` can be learned. This allows us to construct the following LVP, where the first 3 arguments identify the LU, the POS of the LU, and the frame.

```
lvp(buy,verb,'Commerce_buy',[pattern('Buyer','verb->subject',required),
                             pattern('Goods','verb->object',required)]).
```

**Frame-based Parsing.** Once the LVPs are constructed, they can be used to extract logical relations from sentences. Namely, when a new sentence comes in, KALM tries to find the triggered LVPs which are then applied to the sentence to get candidate parses.

**Example 3** Consider the sentence "*A customer buys a watch*," whose DRS is as follows:

```
object(A,customer,countable,na,eq,1)-1/2.   predicate(C,buy,A,B)-1/3.
object(B,watch,countable,na,eq,1)-1/5.
```

The word `buy.verb` triggers the LVP in Example 2. Following the pattern `verb->subject` that extracts the roll-filler of `Buyer`, KALM starts from the LU `buy.verb` (i.e., `predicate(C,buy,A,B)`), and then finds the subject of the LU, which is the 3rd argument of `predicate(C,buy,A,B)` (i.e., the identifier `A`). Finally, the word identified by `A` (i.e., *customer*) is extracted as the role-filler for `Buyer`. In this way, KALM applies all patterns to all extract role-fillers and finally we have the following candidate parse:

```
p('Commerce_buy',[role('Buyer','customer'),role('Goods','watch')]).
```

**Role-Filler Disambiguation.** Generally, a word is associated with several meanings. The goal of role-filler disambiguation is to find the most appropriate sense for each role-filler with respect to the roles in particular logical frames. Role-filler disambiguation is done via a walk through the BabelNet

knowledge graph [10]. BabelNet combines the words with similar meanings into synset nodes. Edges represent semantic relations (hypernym, hyponym, etc.) and strength of the different relationships is specified via weights.

Consider the candidate parse of "*A customer buys a watch*" in Example 3. In BabelNet, the role-filler *watch* has several meanings like "*A small portable timepiece*" (`bn:00077172n`), "*A period of time (4 or 2 hours) during which some of a ship's crew are on duty*" (`bn:00080550n`), and more. Since `Goods` (with the synset `bn:00021045n`) is much more semantically related to a timepiece than to a period of time, *watch* should be disambiguated with the synset `bn:00077172n` denoting a timepiece. Starting from `Goods`'s synset `bn:00021045n`, KALM uses breadth first search to reach *watch*'s synsets `bn:00077172n` and `bn:00080550n` respectively, computes the costs based on the edge weights, and ends up with the synset that has the lowest cost, which is `bn:00077172n` ("*A small portable timepiece*") in this case.

**Constructing Logic Representation.** Ultimately, the disambiguated candidate parses are translated into *unique logical representation* (ULR), which gives the true meaning to the original CNL sentence and is suitable for querying. ULR uses the predicates `frame/2` and `role/2` for representing instances of the frames and the roles. The predicates `synset/2` and `text/2` are used to specify synset and textual information. For example, "*A customer buys a watch*" will be converted into the ULR shown below:

```
frame(id_1,'Commerce_buy').
role(id_1,'Buyer',id_2). synset(id_2,'bn:00019763n'). text(id_2,'customer').
role(id_1,'Goods',id_3). synset(id_3,'bn:00077172n'). text(id_3,'watch').
```

## 3 Factual Sentences

In knowledge authoring, we are not interested in fine letters but rather in sentences that express or query knowledge, such as facts, queries, rules, modalities. In this paper we limit ourselves to facts and queries and more advanced types of knowledge is left to followup papers. Consequently, here we focus on sentences for specifying and querying sets of facts, which we call *factual sentences*. Non-factual sentences, like "*Go fetch more water*," do not express any factual information and can be thus excluded from consideration.

Before defining factual sentences, we first remind some key grammatical concepts. A *clause* is a unit of grammatical organization that contains a verb and usually other components. A *main clause*[3] is a clause that can form a complete sentence standing alone and having a subject and a predicate. A *subordinate clause* depends on a main clause for its meaning. Together with the main clause, a subordinate clause forms part of a *complex sentence*. There are 4 types of subordinate clauses including adnominal clauses, adverbial clauses, clausal complements and clausal subjects. A *coordination* is a syntactic structure that links together two or more elements with connectives such as "*and*" and "*or*" (e.g., *a car and a watch*). When the elements are main clauses, a *compound sentence* is formed (e.g., "*Mary wants the car and the car is available*").

Examination of various datasets shows that main clauses, compound sentences, and sentences with adnominal clauses (e.g. "*Mary bought a car made in USA*") are by far the most common constructs in datasets that contain data and queries. In contrast, clausal complements and other types of subordinate clauses are typically non-factual or they are used to describe other kinds of logical statements, such as rules, which will be the subject of our followup work. For the same reason, connectives other than "*and*"

[3] `https://www.lexico.com/en/definition/main_clause`

and "*or*" are also eliminated. We then define *factual sentence* for knowledge authoring as follows:
**Definition 1** A *factual sentence* is

1. a factual main clause with subordinate adnominal clauses (if any), and no other subordinate clauses; or
2. a compound sentence where the connectives connect only the clauses of the kind described in 1.

**Definition 2** A *main clause is factual*[4] if

- it has a verb with a subject (e.g., "*Mary bought a car*"); or
- it has a nominal word (or an adjective) with a subject and a linking verb (e.g., "*Mary is rich*," where an adjective *rich* has a subject *Mary* and linking verb *is*)

## 3.1 Grammatical Properties of Factual Sentences

We now use POS tags (part of speech) and universal dependencies to describe six *grammatical properties for factual sentences* that follow from the aforesaid factual restriction on the English sentences, and thus are *necessary conditions* for sentences to be factual. We then use these properties to discover and correct errors made by the STANZA parser. We use the superscripts U and X to refer to universal POS[5] (UPOS) tags, Penn Treebank extended POS[6] tags (XPOS), and UD will refer to universal dependency[7] labels.
**Property 1** If the main clause is factual, then

- the main clause has a verb with a subject. That is, the clause has a word with an incoming $\texttt{root}^{\texttt{UD}}$ edge tagged with $\texttt{VERB}^{\texttt{U}}$ and an outgoing $\texttt{nsubj}^{\texttt{UD}}$ edge); or
- the main clause has a nominal word (or an adjective) with a subject and a linking verb. Thus, the clause has a word with an incoming $\texttt{root}^{\texttt{UD}}$ edge that is (i) tagged with $\texttt{NOUN}^{\texttt{U}}$, $\texttt{PRON}^{\texttt{U}}$, $\texttt{PROPN}^{\texttt{U}}$, or $\texttt{ADJ}^{\texttt{U}}$; (ii) has an outgoing $\texttt{nsubj}^{\texttt{UD}}$ edge; and (iii) has an outgoing $\texttt{cop}^{\texttt{UD}}$ edge (copula).

**Property 2** If a word $W$ is the last element of a coordination (e.g., "*watch*" in "*a car or a watch*"), then this coordination must be an *and*- or an *or*-coordination. That is, $W$ has an incoming $\texttt{conj}^{\texttt{UD}}$ edge and a outgoing $\texttt{cc}^{\texttt{UD}}$ edge pointing to "*and*" or "*or*."
**Property 3** If a verb $V$ has one or more auxiliary verbs $V_1^a,...,V_n^a$, and $V_n^a$ (tagged with $\texttt{AUX}^{\texttt{U}}$) is the closest auxiliary verb to $V$ (e.g., in the sentence "*A car has been bought by Mary*," $V_1^a = has$, $V_n^a = V_2^a = been$, $V = bought$), then

- continuous tense ($V_n^a$ is *be* $- V$ is a present participle): $V_n^a$ has an incoming $\texttt{aux}^{\texttt{UD}}$ edge starting from $V$, and $V$ is tagged with $\texttt{VBG}^{\texttt{X}}$; or
- perfect tense ($V_n^a$ is *have* $- V$ is a past participle): $V_n^a$ has an incoming $\texttt{aux}^{\texttt{UD}}$ edge starting from $V$, and $V$ is tagged with $\texttt{VBN}^{\texttt{X}}$; or
- past, present, and future tense ($V_n^a$ is *can/do/may/must/ought/should/will* $- V$ in base form): $V_n^a$ has an incoming $\texttt{aux}^{\texttt{UD}}$ starting at $V$, and $V$ is tagged with $\texttt{VB}^{\texttt{X}}$; or
- passive voice ($V_n^a$ is *be/get* $- V$ is a past participle): $V_n^a$ has an incoming $\texttt{aux:pass}^{\texttt{UD}}$ edge starting from $V$, and $V$ is tagged with $\texttt{VBN}^{\texttt{X}}$

**Property 4** For a verb $V$ without auxiliary verbs (no outgoing $\texttt{aux}^{\texttt{UD}}$/$\texttt{aux:pass}^{\texttt{UD}}$ edges):

[4] According to https://www.lexico.com/en/definition/main_clause, all main clauses are factual.
[5] https://universaldependencies.org/u/pos/
[6] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[7] https://universaldependencies.org/u/dep/

1. if $V$ is a present or past participle (i.e., tagged with $\texttt{VBG}^{\texttt{X}}$/$\texttt{VBN}^{\texttt{X}}$), then
   - $V$ occurs in a coordination (i.e., has an incoming $\texttt{conj}^{\texttt{UD}}$ edge); or
   - $V$ occurs in adnominal clauses (i.e., has an incoming $\texttt{acl}^{\texttt{UD}}$ edge)
2. if $V$ is in present or past tense (i.e., tagged with $\texttt{VBP}^{\texttt{X}}$/$\texttt{VBZ}^{\texttt{X}}$/$\texttt{VBD}^{\texttt{X}}$), then
   - $V$ occurs in a coordination (i.e., has an incoming $\texttt{conj}^{\texttt{UD}}$ edge); or
   - $V$ occurs in main/adnominal clauses (i.e., has an incoming $\texttt{root}^{\texttt{UD}}$/$\texttt{acl}^{\texttt{UD}}$/ $\texttt{acl:relcl}^{\texttt{UD}}$ edge) and have a subject (i.e., an outgoing $\texttt{nsubj}^{\texttt{UD}}$ edge)
3. if $V$ is in the base form (i.e., tagged with $\texttt{VB}^{\texttt{X}}$), then
   - $V$ occurs in a coordination (i.e., has an incoming $\texttt{conj}^{\texttt{UD}}$ edge); or
   - $V$ occurs in adnominal clauses with infinitive form (i.e., has an incoming $\texttt{acl}^{\texttt{UD}}$ edge and an outgoing $\texttt{mark}^{\texttt{UD}}$ edge pointing to "*to*")

**Property 5** If a non-verb word $W$ has one or more auxiliary verbs $V_1^a,...,V_n^a$, where $V_n^a$ is the closest auxiliary verb to $W$ (e.g., in the sentence "*Mary has been rich*," $V_1^a$ =*has*, $V_2^a$ = *been*," $W$ = *rich*, and *been* is the closest auxiliary verb to *rich*), then $V_n^a$ and $W$ must satisfy these properties:

1. $W$ is a nominal word or an adjective (i.e., tagged with $\texttt{NOUN}^{\texttt{U}}$/$\texttt{PRON}^{\texttt{U}}$/$\texttt{PROPN}^{\texttt{U}}$/$\texttt{ADJ}^{\texttt{U}}$)
2. $V_n^a$ is the copula of $W$ (i.e., $V_n^a$ has an incoming $\texttt{cop}^{\texttt{UD}}$ edge starting from $W$)
3. $W$ has a subject (i.e., has an outgoing $\texttt{nsubj}^{\texttt{UD}}$ edge)

**Property 6** The sentence must be *projective*. Given a parse, if there are crossing edges (e.g., the incoming edges for "*of*" and "*Winston*" in Fig. 2), the sentence is called *non-projective*, otherwise it is *projective*. Property 6 expresses the belief held by linguists that well-constructed English sentences are typically projective, and so are factual sentences.

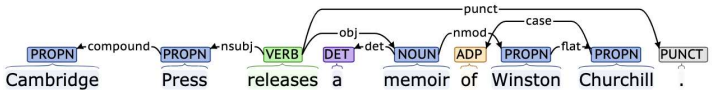



Figure 2: A example of a non-projective parse

## 4 KALM for Factual Language

We now briefly describe a neural parser multi-STANZA (mSTANZA) that generates several ranked parses for input sentences and is a modification of the original STANZA. Then we describe $\text{KALM}^{\text{FL}}$, a product of adaptation of the KALM framework to factual English sentences—a language that is significantly less restricted than any known CNL, and is much easier to learn. The $\text{KALM}^{\text{FL}}$ framework is shown in Fig. 3.

### 4.1 Multi-STANZA

STANZA [11] is a pipelined neural parser with state-of-the-art performance, which was designed to return only the top parse for each sentence. Unfortunately, we found that it frequently errs in POS tagging, and these errors then propagate to universal dependencies. We then noticed that nearly top parses often give correct POS tags where the top parses err and so we modified STANZA to return also some non-top-ranked parses. We called the result mSTANZA. Figure 4 shows the architecture. Each stage adds multiple sets of annotations, creating a new `Document` object for each set. These `Document` objects are then passed downstream. Unlike STANZA, the output of mSTANZA is a list of annotated `Document` objects ranked in the order of decreasing confidence.

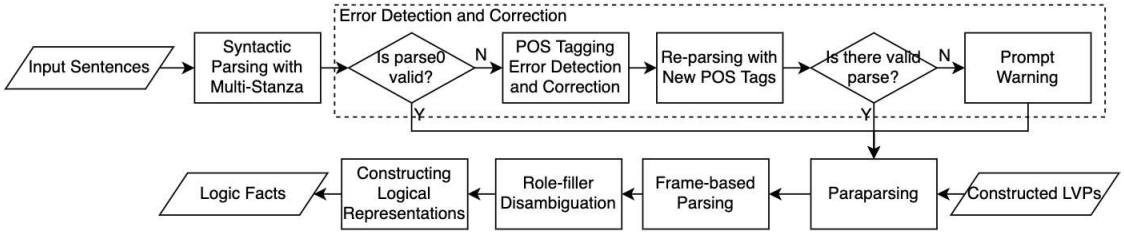



Figure 3: The KALM$^{FL}$ framework

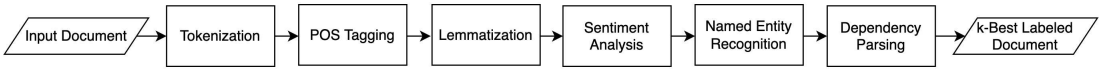



Figure 4: The architecture of mSTANZA

#### 4.1.1 POS Tagger and Dependency Parser

The Part-of-Speech (POS) tagger adds POS tags to each word. As each POS tag has a finite number of categories, it is straightforward to extract the $k$-best POS tags for each word, along with their confidences. mSTANZA allows a user to provide a function to dynamically modify the list of $k$-best POS tags according to their needs. In terms of dependency parsing, mSTANZA generates a dependency parse by generating a fully connected directed graph, and generating the weights of the edges using a neural network. The neural network learns to assign the weight of the edge based on the type of edge and the relationship between the vertices. Then, the minimum spanning arborescence is found and used as the dependency parse. Multiple possible dependency parses for each sentence are combined in order to generate the next-best parse for the entire document.

#### 4.1.2 Error Detection and Correction Based on Multi-STANZA

KALM$^{FL}$ checks STANZA parses for being factual using the necessary conditions of Section 3.1. If any of the checks don't pass, KALM$^{FL}$ attempts to correct the parse by conjecturing that some of the POS tags are wrong (a fairly common problem with STANZA in our experience). This is done by using other nearly top parses provided by mSTANZA. If the correction attempt fails, the user is asked to rephrase the sentence. Details of the error correction algorithm are given in A.

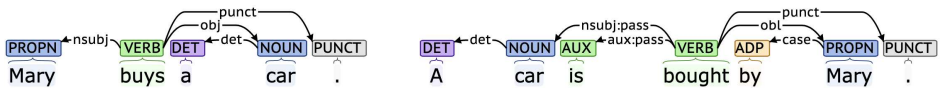
### 4.2 Paraparsing

Paraparsing is a set of corrective steps that modify the original $\overline{parse}$ (see Appendix A). The aim here is to eliminate possible semantic mismatches that were the original motivation for KALM, as explained in the introduction. The mismatches handled here arise from the possibility that the same information may be described via passive or active voice, via a different order of elements in a coordination, via the different ways to attach adnominal clauses, and more. Note that all these corrections became possible in KALM$^{FL}$ due to the use of dependency parsing and were not possible in the original CNL-based KALM.

#### 4.2.1 Passive Voice

mSTANZA handles the active and passive voices separately. For a pair of active/passive voice sentences with the same meaning, such as "*Mary buys a car*" and "*A car is bought by Mary*," STANZA gives

two completely different parses shown in Fig. 5; it does not attempt to reconcile the semantic mismatch between them so that they would yield the same logical representation. To address this problem, KALM$^{FL}$ first recognizes passive voice by the `aux:pass`$^{UD}$ edge in the parse, then modifies the edges of passive voice parses to make the parses equivalent to their active voice counterparts. If the sentence is in active voice, keep the parse unchanged. Otherwise, convert it into *n* parses in active voice (*n* is the number of *by*-phrases in the clause, since every *by*-phrase could be the subject of the real active voice counterpart of this passive voice sentence) by modifying (i) `nsubj:pass`$^{UD}$ to `obl:by` and (ii) *n* `obl:by` edges to `nsubj`$^{UD}$ one by one.

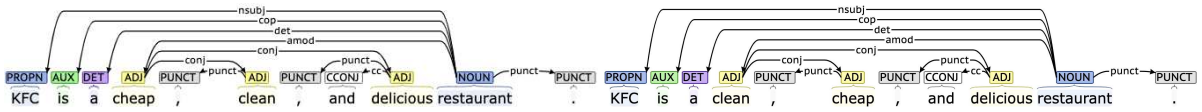



(a) Active voice (b) Passive voice

Figure 5: Semantic mismatch caused by passive voice

### 4.2.2 Coordination

Elements in STANZA coordinations are not treated equally. For example, in the parse of "*KFC is a cheap, clean, and delicious restaurant*" shown in Fig. 6a, *cheap* directly depends on *restaurant*, but *clean* and *delicious* mutually depend on *cheap* instead of *restaurant*. In this case, if *cheap* and *clean* are swapped, the meaning of the sentence stays unchanged, but the parse will be different as shown in Fig. 6b. This phenomenon will lead to a semantic mismatch.




(a) (b)

Figure 6: Semantic mismatch caused by coordination

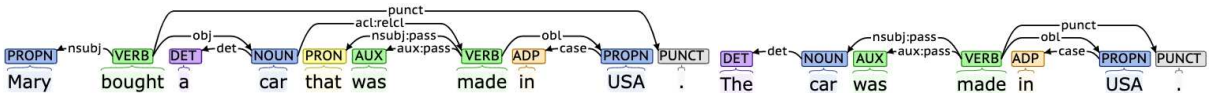
KALM$^{FL}$ treats coordination elements equally by modifying their edges. The procedure is shown below, and all examples refer to Fig. 6a.

1. Locate the root element $el_{root}$ of the coordination. It is a word that has outgoing `conj`$^{UD}$ edges to other elements, which have incoming `conj`$^{UD}$ edges (e.g. $el_{root}$ is *cheap*, while *clean* and *delicious* are the other two elements)
2. Copy the incoming edge of $el_{root}$ to each non-root element and delete the edge `conj`$^{UD}$ (e.g. copy `amod`$^{UD}$ to replace `conj`$^{UD}$ that goes to *clean* and *delicious*)
3. Copy the outgoing edges of $el_{root}$ (other than the deleted `conj`$^{UD}$) to each non-root element (in our example, *cheap* has no outgoing edges, so no need to copy anything)

### 4.2.3 Adnominal Clause

An adnominal clause describes a fact about the nominal word it modifies. For example, "*Mary bought a car that was made in USA*" represents two facts: "Mary bought a car," and "The car was made in USA". However, the second fact is parsed differently when it is in an adnominal clause than when it is in a

sentence by itself, and such phenomena lead to semantic mismatch. As shown in Fig. 7a, the subject of "*a car that was made in USA*" is "*that*" whereas the real subject should be "*car*" like the parse in Fig. 7b.




(a) (b)

Figure 7: Semantic mismatch caused by adnominal clause

KALM$^{FL}$ recognizes the real roles (e.g., subject, object, etc.) that the modified word plays in the adnominal clause, so that the adnominal clause can be seen as a complete sentence all by itself. This is done via the following transformation.

- if a word $V_1$ has an incoming `acl`$^{UD}$ edge $e_1$ that starts at $V_2$ and has no outgoing `nsubj`$^{UD}$ or `nsubj:pass`$^{UD}$ edges, then
  - if $V_1$ is a present participle or a base-form verb tagged with `VBG`$^{X}$ or `VB`$^{X}$, flip the direction of $e_1$ and change the label to `nsubj`$^{UD}$
  - if $V_1$ is a past participle tagged with `VBN`$^{X}$, flip the direction of $e_1$ and change the label to `nsubj:pass`$^{UD}$
- if a word $V_1$ has an incoming `acl:relcl`$^{UD}$ edge $e_1$ that starts at $V_2$, then
  - if $V_1$ has an outgoing `nsubj`$^{UD}$, `nsubj:pass`$^{UD}$ or `obj`$^{UD}$ edge pointing to an introductory word $W_{intro}$ "*that/who/which*," replace $W_{intro}$ with $V_2$
  - if $V_1$ has an outgoing `mark`$^{UD}$ edge pointing to an introductory word $W_{intro}$ *where/when/why/which* replace $W_{intro}$ with $V_2$ and modify `mark`$^{UD}$ to `obl`$^{UD}$

### 4.2.4 Other Semantic Mismatches

Besides the most frequent semantic mismatch issues solved above, KALM$^{FL}$ also tackles other types of semantic mismatch caused by lemmatization, particle verbs, prepositional phrases, named entities, indirect objects, and so forth. As shown in Fig. 3, after the Paraparsing step, the ultimate parse is delivered to Frame-based Parsing and undergoes further processing to ultimately yield a unique disambiguated logical representation.

## 4.3 Representing Dependencies in Logic Programming Systems

The original KALM used DRS to represent extracted information. In this paper, the parses are represented by much more general graphs, so we introduce an appropriate logical representation for them. Here is an example shown below:

**Example 4** The KALM$^{FL}$ representation for the sentence "*Mary buys a car*".

```
token(index(1,1,1),mary,[edge(index(1,2),jbusn)],edge(index(1,2),nsubj),
      propn,nnp,index(1,2),s_person,accepted).
token(index(1,2,1),buy,[edge(index(1,1),nsubj),edge(index(1,4),obj)],edge(index(1,0),root),
      verb,vbz,index(1,2),o,accepted).
token(index(1,3,1),a,[edge(index(1,4),ted)],edge(index(1,4),det),
      det,dt,index(1,2),o,accepted).
token(index(1,4,1),car,[edge(index(1,3),det),edge(index(1,2),jbo)],edge(index(1,2),obj),
      noun,nn,index(1,2),o,accepted).
```

where a sentence is represented by a set of `token/9` predicates and each `token/9` predicate represents a token $t$. The 1st argument in a `token/9` predicate includes sentence ID, candidate parse ID, and $t$'s ID. The 2nd argument is the lemma of $t$. The 3rd argument is a list of edges that connect $t$ to other tokens, and an `edge/2` predicate representing a specific edge $e$ includes the index of the other token on $e$, and the edge type (reversed if it is an in-coming one). The 4th argument is the one and only in-coming edge that $t$ has. The 5th, 6th are $t$'s UPOS and XPOS tags, respectively. The 7th argument is the index of the root token in the whole sentence, namely, "*buys*" in Example 4. Finally, the 8th and 9th arguments are the named entity and validation tags, where the latter indicates if the parse is factual (`accepted`) or not.

## 4.4 Role-filler Disambiguation and Unique Logical Representations

In KALM, a clause represents a complete fact so each clause has only one parse, the one with the highest semantic score after disambiguation. In KALM$^{\text{FL}}$, coordinations and adnominal clauses are introduced and their meanings can be captured accordingly, which is further explained in Section 4.4.1 and 4.4.2.

For logical representations, KALM$^{\text{FL}}$ uses `ulr/3` and `role/4` for representing instances of final parses after disambiguation. Consider theses sentences: "*Mary bought a car for John*" and "*Mary made a purchase of a car for John*". Although they have different syntactic structures, they are ultimately converted into exactly the same parse and their role-fillers are assigned exactly the same synsets. Therefore, they must be translated into a unique logical representation (ULR). And indeed, the ULR for sentences "*Mary bought a car for John*" and "*Mary made a purchase of a car for John*" is the same:

```
ulr(fid_1,'Commerce_buy',[role(rid_1,'Buyer',mary,'bn:00046516n'),
                          role(rid_2,'Goods',car,'bn:00007309n'),
                          role(rid_3,'Recipient',john,'bn:00046516n')]).
```

The first argument `fid_1` is the unique ID of this buying event, the second argument, `'Commerce_buy'`, is the frame name, and argument 3 is a list of role descriptors.

### 4.4.1 ULR for Factual Sentences with Coordinations

Generally, a sentence can have a mixture of *and*- and *or*-coordinations, whose meaning is quite hard to describe. For simplicity, we focus on the case where $C$ has only one type of coordination, i.e., all connectives are *and* or all are *or*.

Let $[C_1,...,C_n]$ be the list of all coordinations in a sentence $S$. A *coordinated choice* is a list $\sigma = [el_1,...,el_n]$ of coordination elements such that $el_i \in C_i$ for all $i = 1,...,n$. Let $S_\sigma$ be $S$ where each coordination, $C_i$, and its elements is replaced with the corresponding element $el_i$ from $\sigma$. Thus, for each coordinated choice $\sigma$ for $S$, the above replacement operation constructs another sentence, $S_\sigma$, which has no coordinations. Next, we collect $S_\sigma$ for all the different $\sigma$'s and organize these sentences as elements of a new homogeneous coordination of the same type as each of the original coordination $C_i$. The result is a sentence $S'$ with only one coordination, found at the root of the parse for the sentence.

For example, for the sentence "*Mary bought and sold a car and a watch*", the coordinated choices includes $\sigma_1 = [bought, car]$, $\sigma_2 = [bought, watch]$, $\sigma_3 = [sold, car]$, $\sigma_4 = [sold, watch]$. Based on the coordinated choices, we can construct 4 sentences without coordinations: "*Mary bought a car*," "*Mary bought a watch*," "*Mary sold a car*," and "*Mary sold a watch*," which are organized into a new *and*-coordination. Thus, we have the final ULR for this *and*-coordination shown below:

```
ulr(fid_1,'Commerce_buy',[role(rid_1,'Buyer',mary,'bn:00046516n'),
                          role(rid_2,'Goods',car,'bn:00007309n')]).
ulr(fid_2,'Commerce_buy',[role(rid_1,'Buyer',mary,'bn:00046516n'),
```

```
                          role(rid_3,'Goods',watch,'bn:00077172n')]).
ulr(fid_3,'Commerce_sell',[role(rid_1,'Seller',mary,'bn:00046516n'),
                          role(rid_2,'Goods',car,'bn:00007309n')]).
ulr(fid_4,'Commerce_sell',[role(rid_1,'Seller',mary,'bn:00046516n'),
                          role(rid_3,'Goods',watch,'bn:00077172n')]).
```

#### 4.4.2 ULR for Factual Sentences with Adnominal Clauses

An adnominal clause always describes the nominal word it modifies. This means that an adnominal clause expresses a complete fact about the nominal word. In other words, the facts represented by adnominal clauses and by the main clause must both hold. Thus, the ULRs for clauses must be in conjunction. For example, the sentence "*[Mary bought a car]$_{main}$ [made in the country]$_{adnominal}$ [that John lives in]$_{adnominal}$*" has a main clause and two adnominal clauses, one modifying the word "*car*" and the other the word "*country*." The ULR then is given below:

```
ulr(fid_1,'Commerce_buy',[role(rid_1,'Buyer',mary,'bn:00046516n'),
                          role(rid_2,'Goods',car,'bn:00007309n')]).
ulr(fid_2,'Manufacturing',[role(rid_2,'Product',car,'bn:00007309n'),
                           role(rid_3,'Place',country,'bn:00023236n')]).
ulr(fid_3,'Residence',[role(rid_4,'Resident',john,'bn:00046516n'),
                       role(rid_3,'Location',country,'bn:00023236n')]).
```

## 5 Evaluation

We use four datasets to demonstrate the high performance of KALM$^{\text{FL}}$ as a knowledge authoring machine for factual English.

### 5.1 Datasets

- **CNLD.** [8] uses CNL sentences, largely inspired by FrameNet, to evaluate the original KALM. We call this dataset the CNL Dataset (CNLD). CNLD contains 250 short CNL sentences in present tense, such as "*Kate purchases a house*." This dataset is captured via 50 logical frames and 317 LVPs constructed from 213 training sentences.
- **CNLDM.** This dataset is obtained from CNLD by changing the voice of some sentences from active to passive and vice versa. In addition, some sentences are changed to past or future tense. Thus, CNLDM contains sentences like "*A house was purchased by Kate*," with mixed voice and tense. Our evaluation uses the same LVPs as in CNLD.
- **MetaQA.** This dataset [14] has queries that use complex adnominal clauses. These queries neatly fall into several different templates. Within each template, the queries differ only in the entity names. Also, all named entities are pre-annotated. For example, the queries "*who directed the movies written by [Thomas Ian Griffith]*" and "*who directed the movies written by [Frank De Felitta]*" belong to the same template "*who directed the movies written by [MASK],*" where *[MASK]* is a placeholder for pre-annotated named entities. In this evaluation, we use 2- and 3-hop templates directly instead of the original queries, because with named entities annotated, different queries that fall into a same template have exactly the same mSTANZA parse. Only 3 frames are needed to represent the semantics of all such queries: `Movie`, `Inequality`, and `Coop`(eration). Acting as knowledge engineers, we designed 85 training sentences and used the approach of [9] to understand 2- and 3-hop queries.

- **NLD.** NLD uses part of the dataset from FrameNet. NLD includes 250 sentences which look like: "*GDA has purchased the site from Laing Homes and plans are being prepared for an 80 million dollar mixed development for business, media and leisure activities*", which is the original sentence of the CNLD sentence "*Kate purchases a house*". NLD sentences have much more complicated structures that goes beyond factual sentences. Besides factual parts, most of the NLD sentences have non-factual parts that are not usable for knowledge acquisition. In view of this, we ignore all the non-factual parts in NLD sentences. Another approach could be highlighting the non-factual parts and letting the user to correct them or eliminate them.

For the original KALM, all these datasets have to be manually modified to eliminate future/past tense, to put adnominal clauses in a certain canonical form, restrict the vocabulary for the controlled natural language parser, particle verbs, appositives and compound nouns also had to be manually modified. In KALM$^{FL}$, all this is done automatically, and therefore, it can handle a much bigger share of natural language sentences.

## 5.2 Comparison Systems

We compare KALM$^{FL}$ with the original KALM as well as three other frame-based parsers: SEMAFOR [2], SLING [12], and OpenSesame [13]. SEMAFOR and SLING have been previously shown to be inaccurate in [7], so we will not repeat these findings and instead focus on the recently proposed OpenSesame system. Unlike KALM$^{FL}$, OpenSesame is a three-staged pipeline involving target (i.e., LU) identification, frame identification and argument (i.e., role-filler) identification—each stage is essentially a neural network trained independently of the others. In addition, we consider the neural system DrKIT [3] as another comparison system, which achieves the best performance on MetaQA among neural models.

## 5.3 Results

The evaluation is based on the following metrics:

1. **Frame-level Micro-F1**: the ratio of sentences that (i) correctly trigger all the applicable frames, and (ii) do not trigger wrong frames.
2. **Role-level Micro-F1**: the ratio of sentences that (i) correctly trigger all the applicable frames with all roles correctly identified, and (ii) do not trigger wrong frames.
3. **Synset-level Micro-F1**: the ratio of sentences that (i) correctly trigger all the applicable frames with all roles correctly identified and disambiguated, and (ii) do not trigger wrong frames. Note this metric applies only to KALM and KALM$^{FL}$; other systems do not attempt to give semantics with this level of precision.

Table 1: Micro-F1 score comparisons on different datasets

| | CNLD | | | CNLDM | | | MetaQA | | | NLD | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F | R | S | F | R | S | F | R | S | F | R | S |
| KALM | 0.99 | 0.99 | 0.97 | – | – | – | **1.00** | **1.00** | **1.00** | – | – | – |
| OpenSesame | 0.61 | 0.17 | – | 0.59 | 0.11 | – | 0.49 | 0.00 | – | 0.56 | 0.12 | – |
| KALM$^{FL}$ | **0.99** | **0.99** | **0.97** | **0.99** | **0.99** | **0.97** | 0.95 | 0.95 | 0.95 | **0.99** | **0.98** | **0.95** |
| DrKIT | – | – | – | – | – | – | – | 0.876 | – | – | – | – |

Results for the different levels of F1 scores are presented in Table 1. In the table, F, R, and S refer to the frame, role, and synset-level F1 scores, respectively. Note that the results reported in the literature for DrKIT [3] can be interpreted as pertaining the role-level Micro-F1 metric. The results in the table are summarized below.

1. CNLD: This dataset has only CNL sentences and KALM$^{FL}$ achieves the same high F1 scores as the original KALM in all metric levels. OpenSesame's 0.61 frame-level F1 score shows that it has difficulty even to recognize correct frames.
2. CNLDM: Perturbation of the tenses and voices of CNLD sentences took this dataset outside of the Attempto's APE CNL, thus the original KALM cannot handle some of the CNLDM sentences even though the meaning of these sentence did not change.
3. MetaQA: KALM performs perfectly, but only after changing the sentences so they comply with the ACE CNL. In contrast, KALM$^{FL}$ gets the 0.95 synset F1 score even without any preprocessing. OpenSesame fails on MetaQA with 0 role-level F1 score—probably because it was never trained on the movie domain. In the comparison between DrKIT and KALM$^{FL}$, DrKIT [3] achieved 0.871 and 0.876 accuracy on 2- and 3-hop query answering respectively. For KALM$^{FL}$, the 333 out of 350 correctly parsed templates covers 128,784 2-hop queries and 119,923 3-hop queries, which results in 0.962 and 0.933 accuracy on 2- and 3-hop query answering and outperforms DrKIT.
4. NLD: The original KALM fails since this dataset breaks the CNL restrictions on the input language. In contrast, KALM$^{FL}$ does well and easily outperforms OpenSesame, especially when it comes to handling of adnominal clauses.

It is surprising that OpenSesame's role level F1 scores are extremely low on the three FrameNet-related datasets. Error analysis shows that even for simple CNLD sentences like "*Mary buys a laptop*," OpenSesame has hard time extracting all roles. For instance, "*laptop*" is not extracted as a role-filler for the role `Goods`.

## 6 Limitations and Future Work

Although KALM$^{FL}$ has been shown to have high accuracy, limitations still exist. For example,

- KALM$^{FL}$ accepts various tenses based on Property 3, but currently this and other temporal information is ignored and is planned for future work.
- KALM$^{FL}$ doesn't handle anaphora because the quality of the parses is highly dependent on the quality of the chosen off-the-shelf anaphora resolver.
- KALM$^{FL}$ treats sentences with quantifiers, like "*Every pet has an owner*," as facts rather than rules, which points to an issue with the definition of factual sentences.

For future work, we plan to address some of the aforesaid problems and to extend KALM$^{FL}$ with quantifiers, rules, temporal information, and other advanced features that have direct counterparts in natural languages.

## 7 Conclusion

The original KALM [7, 8, 9] was proposed as a solution to the problem of semantic mismatch in knowledge authoring using natural languages, but this solution was limited to CNLs, which is a severe limitation both in expressiveness and human training. In this paper, we introduced KALM$^{FL}$, an NLP system

that is not chained by CNL limitations. The only restriction is that the sentences used for knowledge authoring must be factual, i.e., express factual information as opposed to, say, feelings, allegories, hyperbolas, etc. Benchmarking shows that this approach captures the meanings of factual sentences with very high accuracy: the 0.95 F1 score for both facts and queries.

## References


[1] Collin F Baker, Charles J Fillmore & John B Lowe (1998): *The berkeley framenet project*. In: *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pp. 86–90, doi:10.3115/980845.980860.

[2] Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider & Noah A Smith (2014): *Frame-semantic parsing*. *Computational linguistics* 40(1), pp. 9–56, doi:10.1162/COLI_a_00163.

[3] Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov & William W Cohen (2020): *Differentiable reasoning over a virtual knowledge base*. *arXiv preprint arXiv:2002.10640* 1, doi:10.48550/arXiv.2002.10640.

[4] Charles J Fillmore et al. (2006): *Frame semantics*. *Cognitive linguistics: Basic readings* 34, pp. 373–400, doi:10.1515/9783110199901.373.

[5] Norbert E Fuchs, Stefan Hoefler, Kaarel Kaljurand, Tobias Kuhn, Gerold Schneider & Uta Schwertel (2006): *Discourse Representation Structures for ACE 5*. *ifi Technical Reports* 1(ifi2006. 10), doi:10.5167/uzh-62058.

[6] Norbert E Fuchs & Rolf Schwitter (1996): *Attempto controlled english (ace)*. *arXiv preprint cmp-lg/9603003* 1, doi:10.48550/arXiv.cmp-lg/9603003.

[7] Tiantian Gao, Paul Fodor & Michael Kifer (2018): *High accuracy question answering via hybrid controlled natural language*. In: *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, IEEE, pp. 17–24, doi:10.1109/WI.2018.0-112.

[8] Tiantian Gao, Paul Fodor & Michael Kifer (2018): *Knowledge authoring for rule-based reasoning*. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, pp. 461–480, doi:10.1007/978-3-030-02671-4_28.

[9] Tiantian Gao, Paul Fodor & Michael Kifer (2019): *Querying Knowledge via Multi-Hop English Questions*. *Theory and Practice of Logic Programming* 19(5-6), pp. 636–653, doi:10.1017/S1471068419000103.

[10] Roberto Navigli & Simone Paolo Ponzetto (2012): *BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network*. *Artificial Intelligence* 193, pp. 217–250, doi:10.1016/j.artint.2012.07.001.

[11] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton & Christopher D Manning (2020): *Stanza: A python natural language processing toolkit for many human languages*. *arXiv preprint arXiv:2003.07082* 1, doi:10.18653/v1/2020.acl-demos.14.

[12] Michael Ringgaard, Rahul Gupta & Fernando CN Pereira (2017): *SLING: A framework for frame semantic parsing*. *arXiv preprint arXiv:1710.07032* 1, doi:10.48550/arXiv.1710.07032.

[13] Swabha Swayamdipta, Sam Thomson, Chris Dyer & Noah A Smith (2017): *Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold*. *arXiv preprint arXiv:1706.09528* 1, doi:10.48550/arXiv.1706.09528.

[14] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola & Le Song (2018): *Variational reasoning for question answering with knowledge graph*. In: *Thirty-second AAAI conference on artificial intelligence*, pp. 6069–6076, doi:10.48550/arXiv.1709.04071.

# A Error Detection and Correction

Fig. 8 illustrates one of the errors in STANZA POS tagging. Here, the word *protests* is wrongly tagged as a noun and the dependencies related to *protests* are also wrong.

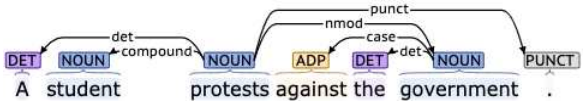



Figure 8: The best mSTANZA parse for "*A student protests against the government*"

Fortunately, the six properties that stem from the factual sentence requirement can detect and help correct some of these mis-taggings. Denote the mSTANZA parse with the highest confidence score as $parse_0 = [w_1, ..., w_n]$ where $n$ is the number of the words, $w_i$ contains all parsing information such as POS tag, dependency relation of the $i$-th word in the sentence. Let $Upos = [upos_1, ..., upos_n]$ and $Xpos = [xpos_1, ..., xpos_n]$ be the UPOS and XPOS taggings of the sentence.

**Algorithm 1** POS Tagging Error Detection and Correction

**Input:** $upos_i$, $xpos_i$, the second best UPOS tag $upos'_i$ for $w_i$, and the second best XPOS tag $upos'_i$ for $w_i$.
**Output:** The corrected UPOS $\overline{upos_i}$ and XPOS $\overline{xpos_i}$ for $w_i$.

$\overline{upos_i}, \overline{xpos_i} \leftarrow upos_i, xpos_i$
**if** $upos_i.score < 0.9$ **then**
  **if** $upos_i ==$ $\texttt{NOUN}^{\texttt{U}}$ and $upos'_i ==$ $\texttt{VERB}^{\texttt{U}}$ **then**
    $\overline{upos_i} \leftarrow \texttt{VERB}^{\texttt{U}}$
    $\overline{xpos_i} \leftarrow \texttt{VBP}^{\texttt{X}}/\texttt{VBZ}^{\texttt{X}}/\texttt{VBD}^{\texttt{X}}$
  **else if** $upos_i ==$ $\texttt{VERB}^{\texttt{U}}$ and $upos'_i ==$ $\texttt{AUX}^{\texttt{U}}$ **then**
    $\overline{upos_i} \leftarrow \texttt{AUX}^{\texttt{U}}$
    $\overline{xpos_i} \leftarrow \texttt{VBP}^{\texttt{X}}/\texttt{VBZ}^{\texttt{X}}/\texttt{VBD}^{\texttt{X}}$
  **else if** $upos_i ==$ $\texttt{PRON}^{\texttt{U}}$ and $upos'_i ==$ $\texttt{DET}^{\texttt{U}}$ **then**
    $\overline{upos_i} \leftarrow \texttt{DET}^{\texttt{U}}$
    $\overline{xpos_i} \leftarrow \texttt{WDT}^{\texttt{X}}/\texttt{PDT}^{\texttt{X}}/\texttt{DT}^{\texttt{X}}$
  **else if** $upos_i ==$ $\texttt{SCONJ}^{\texttt{U}}$ and $upos'_i ==$ $\texttt{ADV}^{\texttt{U}}$ **then**
    $\overline{upos_i} \leftarrow \texttt{ADV}^{\texttt{U}}$
    $\overline{xpos_i} \leftarrow \texttt{WRB}^{\texttt{X}}/\texttt{IN}^{\texttt{X}}$
**else if** $xpos_i.score < 0.9$ **then**
  **if** $xpos_i ==$ $\texttt{VBD}^{\texttt{X}}$ and $xpos'_i ==$ $\texttt{VBN}^{\texttt{X}}$ **then**
    $\overline{xpos_i} \leftarrow \texttt{VBN}^{\texttt{X}}$
  **else if** $xpos_i ==$ $\texttt{VBN}^{\texttt{X}}$ and $xpos'_i ==$ xposVBD **then**
    $\overline{xpos_i} \leftarrow \texttt{VBD}^{\texttt{X}}$
  **else if** $xpos_{1j} ==$ $\texttt{VBP}^{\texttt{X}}$ and $xpos'_i ==$ $\texttt{VB}^{\texttt{X}}$ **then**
    $\overline{xpos_i} \leftarrow \texttt{VB}^{\texttt{X}}$
**return** $\overline{upos_i}, \overline{xpos_i}$

**Detection and correction of POS tags.** As shown in the dotted box of Fig. 3, this step starts with $parse_0$. If $parse_0$ satisfies all the above-mentioned factual properties, $parse_0$ is assumed to be error-free and is directly sent to the Paraparsing step. Otherwise, following Algorithm 1, if a possibly

wrong (with confidence $< 0.9$) POS tag belongs to a certain type of frequent POS tagging errors (e.g., mSTANZA relatively frequently mis-tags verbs as nouns, and this is what happened with *protests* in Fig. 8), KALM$^{\text{FL}}$ then asserts the tag is wrong and corrects it. Lines 2 and 3 in Algorithm 1 capture such type of errors and assign corrected POS tags. Note that in Lines 5, 8, 11, and 14, the algorithm faces multiple options like $\texttt{VBP}^{\texttt{X}}/\texttt{VBZ}^{\texttt{X}}/\texttt{VBD}^{\texttt{X}}$ and chooses the one with the highest confidence score.

**Re-parsing with new POS tags.** Having re-tagged the words in the above step, the new POS tags, $\overline{Upos} = [\overline{upos_1}, ..., \overline{upos_n}]$ and $\overline{Xpos} = [\overline{xpos_1}, ..., \overline{xpos_n}]$, are fed to the mSTANZA dependency parser to re-generate a new dependency parse $\overline{Parse}$, ranked by confidence scores.

**Selecting a corrected parse.** In this step, KALM$^{\text{FL}}$ goes through the parses in $\overline{Parse}$, from the highest confidence score to lowest, looking for a parse, $parse'$, that satisfies all the properties of factual sentences. If $parse'$ is found, it is taken as a corrected parse, $\overline{parse} = parse'$. If $parse'$ is not found, the algorithm assumes the sentence in question is not factual, so it asks the user to paraphrase the sentence.