Great Service! Fine-grained Parsing of Implicit Arguments

Cui, Ruixiang; Hershcovich, Daniel

Published in:
Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)

DOI:
10.18653/v1/2021.iwpt-1.7

Publication date:
2021

Document version
Publisher’s PDF, also known as Version of record

Document license:
CC BY

Citation for published version (APA):
Great Service! Fine-grained Parsing of Implicit Arguments

Ruixiang Cui and Daniel Hershcovich
Department of Computer Science
University of Copenhagen
{rc, dh}@di.ku.dk

Abstract

Broad-coverage meaning representations in NLP mostly focus on explicitly expressed content. More importantly, the scarcity of datasets annotating diverse implicit roles limits empirical studies into their linguistic nuances. For example, in the web review “Great service!”, the provider and consumer are implicit arguments of different types. We examine an annotated corpus of fine-grained implicit arguments (Cui and Hershcovich, 2020) by carefully re-annotating it, resolving several inconsistencies. Subsequently, we present the first transition-based neural parser that can handle implicit arguments dynamically, and experiment with two different transition systems on the improved dataset. We find that certain types of implicit arguments are more difficult to parse than others and that the simpler system is more accurate in recovering implicit arguments, despite having a lower overall parsing score, attesting current reasoning limitations of NLP models. This work will facilitate a better understanding of implicit and underspecified language, by incorporating it holistically into meaning representations.

1 Introduction

Studies of form and meaning, the dual perspectives of the language sign, can be traced back to modern linguistics since de Saussure (1916, 1978). They are relevant to modern NLP, as even current large neural language models cannot intrinsically achieve a human-analogous understanding of natural language (Žabokrtský et al., 2020; Bender and Koller, 2020). Computational linguists attempt to capture syntactic and semantic features by means of constructing meaning representation frameworks, such as PTG (Böhmová et al., 2003), EDS (Oepen and Lønning, 2006), AMR (Banarescu et al., 2013) and UCCA (Abend and Rappoport, 2013). Through such frameworks, researchers have been exploring linguistic phenomena such as quantification (Pustejovský et al., 2019), coreference (Prange et al., 2019), and word sense (Schneider et al., 2018).

In recent years, a few studies have been dedicated to annotating and modelling implicit and underspecified language (Roesiger et al., 2018; Elazar and Goldberg, 2019; McMahan and Stone, 2020). For example, in the online review “there is no delivery”, two of the omitted elements are the business agent and delivered items. However, previous works focus on specific phenomena requiring linguistic and collaborative reasoning, e.g., bridging resolution, numeric fused-heads identification and referential communication. Such datasets overexpose models to limited linguistic expressions, without leveraging the complete syntactic and semantic features of the context, regardless of the diversity of implicit roles.

O’Gorman (2019) and Cui and Hershcovich
present works on fine-grained implicit role typology, incorporating it into the meaning representation frameworks AMR and UCCA, respectively. They lay a foundation for the interpretation of idiosyncratic behaviours of implicit arguments from a linguistic and cognitive perspective. Nevertheless, neither provided a dataset ready for computational studies of such implicit arguments.

We take the latter as a starting point, addressing several theoretical inconsistencies, and evaluate its applicability by carefully re-annotating their pilot dataset and providing inter-annotator agreement. Our categorisation set, consisting of six implicit role types, is compatible with UCCA’s semantic notion of Scene rather than specific linguistic phenomena. Furthermore, as opposed to previous work, it tackles only essential implicit arguments, salient in cognitive processing.

We design the first semantic parser, with two different transition systems, that has the ability to parse fine-grained implicit arguments for meaning representations and evaluate its performance on the revisited dataset. To conclude, we reflect on this work’s objectives and the challenges to face in future research on implicit arguments.

2 Revisiting Implicit Argument Refinement

Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), a typologically-motivated meaning representation framework, has been targeted by several parsing shared tasks (Hershcovich et al., 2019b; Oepen et al., 2019, 2020). It uses directed acyclic graphs (DAG) anchored in surface tokens, where labelled edges represent semantic relations. In the foundational layer, these are based on the notion of Scenes (States and Processes), their Participants and modifiers. UCCA distinguishes primary edges, corresponding to explicit content, from remote edges, allowing cross-Scene relations. Additionally, Implicit units represent entities of importance to the interpretation of a Scene that are not explicitly anchored in the text. Several refinement layers have been proposed beyond the foundational layer, adding distinctions important for semantic roles and coreference (Shalev et al., 2019; Prange et al., 2019).

2.1 Fine-grained Implicit Argument Refinement for UCCA

Cui and Hershcovich (2020) proposed a fine-grained implicit argument typology implemented as a refinement layer of Participants for UCCA, centering around the semantic notion of Scene. Their proposed categorisation set consisting of six types, listed in Table 1, is argued to have low annotation complexity and ambiguity, thus requiring relatively low cognitive load for annotation than other fine-grained implicit argument typologies (O’Gorman, 2019).

For example, the online review “Great service and awesome price!” is annotated as follows and visualised as Figure 1:

(1) \( [\text{Great}_{D}\text{ service}_{P}\text{ IMP}_{\text{Generic}}\text{ IMP}_{\text{Genre-based}}]_{H} \) and \( [\text{awesome}_{S}\text{ prices}_{A}\text{ IMP}_{\text{Genre-based}}]_{!} \)

There are two Scenes invoked in the sentence. “Great service” with “service” as a Process, and “awesome prices” with “awesome” as a State. Who is serving, who is being served and what is priced are distinct implicit arguments, which require reasoning to resolve.

Genre-based roles refer to conventional omission in the genre (Ruppenhofer and Michaelis, 2010). The corpus is based on online reviews, where reviewers typically do not mention what is under review. Therefore, an implicit argument in each Scene is marked as Genre-based referring to the reviewee. Generic roles denote “people in general” (Lambrecht and Lemoine, 2005). In the example, the recipient could be anyone, rather than a specific person.

2.2 Revisiting Inconsistencies

Despite the general soundness of Cui and Hershcovich (2020)’s typology, we find multiple inconsistencies in it: prominently, the treatment of Process and State Scenes and of nominalisation, and some borderline cases. We propose to revisit these cases and introduce consistent implicit argument refinement guidelines.

2.2.1 State Scenes and Process Scenes

UCCA differentiates Scenes according to their persistency in time (Abend and Rappoport, 2013): static Scenes are temporally persistent states, while processual Scenes are evolving events. Cui and Hershcovich (2020) did not annotate implicit arguments in State Scenes, although they are essentially...
While UCCA only annotates implicit units when they are “prominent in the interpretation of the Scene” (Abend and Rappoport, 2013), it is not always clear what should be regarded as such: the level of uniqueness plays an essential role in recognizing the prominence of an argument. For example, even distinguishing definite and indefinite articles may pose a challenge to semantic analysis (Carlson and Sussman, 2005). There are some linguistic tests for testing whether an argument is semantically mandatory (Goldberg, 2001; Hajič et al., 2012). As for UCCA, we posit that time, location and instrument modifiers, as a rule, are ubiquitous to such an extent that no implicit argument should be annotated unless a Process or State warrants it. In Example 3 (visualised as the upper graph in Figure 2a), in the Scene “you leave,” the departing action demands that the implicit source location is vital for understanding, unlike the location element in Example 1 or 2, where they are of low prominence.

2.2.2 The Definition of Prominent Elements

While UCCA only annotates implicit units when they are “prominent in the interpretation of the Scene” (Abend and Rappoport, 2013), it is not always clear what should be regarded as such: the level of uniqueness plays an essential role in recognizing the prominence of an argument. For example, even distinguishing definite and indefinite articles may pose a challenge to semantic analysis (Carlson and Sussman, 2005). There are some linguistic tests for testing whether an argument is semantically mandatory (Goldberg, 2001; Hajič et al., 2012). As for UCCA, we posit that time, location and instrument modifiers, as a rule, are ubiquitous to such an extent that no implicit argument should be annotated unless a Process or State warrants it. In Example 3 (visualised as the upper graph in Figure 2a), in the Scene “you leave,” the departing action demands that the implicit source location is vital for understanding, unlike the location element in Example 1 or 2, where they are of low prominence.

2.2.4 Indefinite Deictic Participant

Deictic arguments refer to the implicit speaker or addressee. They may be confused with Generic roles when it is ambiguous whether the speaker or addressee participates in the Scene. As a rule, we posit that an implicit Participant ought to be Generic unless it evidently refers to the speaker or addressee. Example 4 shows an ambiguous case.

3 Fine-grained Implicit Argument Corpus

As a pilot annotation experiment, Cui and Hershcovich (2020) reviewed and refined 116 randomly selected passages from an UCCA-annotated dataset of web reviews (UCCA EWT; Hershcovich et al., 2019a). The dataset was annotated by only one annotator. With our revisited guidelines, we ask two annotators\(^2\) to revisit the original dataset, adding or modifying implicit arguments, and subsequently refining their categories, using UCCAApp (Abend et al., 2017).

3.1 Evaluation of Implicit Argument Annotation

Standard UCCA evaluation compares two graphs (e.g., created by different annotators, or one being the gold annotation and the other predicted by a parser), providing an F1 score by matching the edges by their terminal span and category (Hershcovich et al., 2019b). However, the standard evaluation completely ignores implicit argument annotation. To quantify inter-annotator agreement and later parser performance (§6), we provide an evaluation metric taking these units into account.

To compare a graph \(G_1\) with implicit units \(I_1\) to a graph \(G_2\) with implicit units \(I_2\) over the same sequence of terminals \(W = w_1, \ldots, w_m\), for each implicit node \(i\), we identify its parent node \(p(i)\), denoting the set of terminals spanned by it as the yield \(y(p(i)) \subseteq W\), and its category as the label \(\ell(p(i), i)\). Define the set of mutual implicit units between \(G_1\) and \(G_2\):

\[
M(G_1, I_1, G_2, I_2) = \left\{ (i, j) \in I_1 \times I_2 \mid \begin{array}{c}
y_1(p_1(i)) = y_2(p_2(j)) \land \\
\ell_1(p_1(i), i) = \ell_2(p_2(j), j)
\end{array} \right\}
\]

\(^2\)The annotators have a background in linguistics and cognitive science. One has experience in annotating UCCA’s Foundational Layer. Training for the other took ~10 hours.
The F-score is the harmonic mean of labelled precision and recall, defined by dividing $|M(G_1, I_1, G_2, I_2)|$ by $|I_1|$ and $|I_2|$, respectively.

In labelled evaluation, it is worth noting that we require a full match of the two sets of labels/categories rather than just intersection, which suffices in standard UCCA evaluation (of non-implicit units). That is, if there are two or more implicit nodes under one parent, it is only considered a correct match when both the numbers of implicit nodes and their labels are equal. We also introduce unlabelled evaluation, which only requires that parents’ spans match.

For example, in Figure 2a, the reference graph has two implicit arguments. The first implicit unit’s parent spans {have, a, real, mechanic, check} and the second’s {you, leave}. In the predicted graph, two implicit units are predicted. The first implicit unit’s parent spans {a, real, mechanic} while the second one spans the same terminals as the reference graph. We can see that the spans of the first implicit unit do not match—the second matches but with the wrong label. Therefore, in labelled and unlabelled implicit unit evaluation, the precision, recall, and F1 scores are all 0 and 0.5, respectively.

In Figure 2b, the reference graph has three implicit units, labelled Non-specific, Generic and Genre-based. One spans {The, service, is, over-rated} while two span {The, service}. Although the predicted graph has two implicit units with correct labels (Non-specific and Generic), it misses both implicit units under the second parent span,
3.2 Inter-annotator Agreement

The two annotators separately reviewed and refined 15 out of the 116 passages (taken from the original test split). Using the evaluation metric proposed in §3.1, the labelled and unlabelled F1 scores are 73.8% and 91.3% respectively (see Appendix A). Annotators have a Cohen’s κ (Cohen, 1960) of 69.3% on the six-type fine-grained classification of the implicit arguments whose parents’ spans match. For comparison, on the FiGref scheme, O’Gorman (2019) report a Cohen’s κ of 55.2% on a 14-way classification and of 58.1% on a four-way classification. Gerber and Chai (2010) proposed a relevant task to annotate implicit arguments for instances of nominal predicates in sentences, which has a Cohen’s κ of 67%. While we maintain a comprehensive fine-grained typology, we still see an improvement in agreement over other corpora.

3.3 Statistics of Revisited Implicit Corpus

Finally, one annotator reviewed and refined all 116 passages. The second reviewed their annotation after completion. The full revisited dataset contains 393 passages, 3700 tokens and 5475 nodes. Table 2 compares our revisited dataset to the unreviewed dataset of Cui and Hershcovich (2020).

<table>
<thead>
<tr>
<th></th>
<th>Deictic</th>
<th>Generic</th>
<th>Genre-based</th>
<th>Type-identifiable</th>
<th>Non-specific</th>
<th>Iterated-set</th>
<th>Total Implicit Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oepen et al. (2019)</td>
<td>66</td>
<td>65</td>
<td>103</td>
<td>39</td>
<td>100</td>
<td>12</td>
<td>385</td>
</tr>
<tr>
<td>Our dataset</td>
<td>107</td>
<td>86</td>
<td>147</td>
<td>6</td>
<td>36</td>
<td>9</td>
<td>391</td>
</tr>
</tbody>
</table>

Table 2: Statistics of Revisited Implicit Corpus compared to the pilot annotation.

viz. the two sets merely intersect but are not equal. Therefore, the triplet precision, recall and F1 score for labelled evaluation are all 0.5 and the triplet for unlabelled evaluation are 1.

4 Two Transition Systems for Parsing Implicit Arguments

We build the first neural parser that supports parsing implicit arguments dynamically in meaning representations, with two different transition systems. We design a transition-based parser, modelled upon Nivre (2003): a stack $S = (\ldots, s_1, s_0)$ holds processed words. $B = (b_0, b_1, \ldots)$ is a buffer containing tokens or nodes to be processed. $V$ is a set of nodes, and $E$ is a set of labelled edges. We denote $s_0$ as the first element on $S$ and $b_0$ as the first element on $B$. Given a sentence composed by a sequence of tokens $t_1, t_2, \ldots, t_n$, the parser is initialized to have a Root node on $S$, and all surface tokens in $B$. The parser will at each step deterministically choose the most probable transition based on its current parsing state. Oracle action sequences are generated for training on gold-standard annotations.

We propose two transition systems, IMPLICIT-EAGER and IMPLICIT-STANDARD, to deal with implicit arguments over the architecture of HIT-SCIR 2019 (Che et al., 2019), which ranked first in UCCA parsing in the MRP 2019 shared task (Oepen et al., 2019). The transition system incorporates all nine transitions, namely, LEFT-EDGE, RIGHT-EDGE, SHIFT, REDUCE, NODE, SWAP, LEFT-REMOTE, RIGHT-REMOTE and FINISH.

SHIFT, together with REDUCE, are standard transitions. SHIFT moves $b_0$ to $S$, while REDUCE pops $s_0$ from $S$ (when it should not be attached to any element in $B$).

Following transition-based constituent parsing, NODE$_X$ creates a new non-terminal node (Sagae and Lavie, 2005). Such node will be created on the buffer, as a parent of $s_0$ with an X-labelled edge.

LEFT-EDGE$_X$ and RIGHT-EDGE$_X$ add an X-labelled primary edge between the first two elements on $S$. When the first element is the parent of the second element on $S$, LEFT-EDGE$_X$ is executed; in reverse, RIGHT-EDGE$_X$ will be chosen when the second element has the first element as its child. The left/right direction is the same as...
Table 3: The transition sets of two implicit transition systems. Actions marked in red are for IMPLICIT-EAGER, blue for IMPLICIT-STANDARD. We write the stack with its top to the right and the buffer with its head to the left. \((\cdot, \cdot)_X\) denotes a X-labelled edge, \((\cdot, \cdot)^*_{X}\) a remote X-labelled edge, and \((\cdot, \cdot)^\#_{X}\) an X-labelled edge to an implicit node. \(i(x)\) is a running index for the created nodes. The prospective child of the EDGE action cannot have a primary parent. The newly generated node by IMPLICIT\(_X\) action is prohibited from having any descendant. NODE generates a concept node on the buffer, but does not produce an arc. This table is adapted from Hershcovich et al. (2017).

where the arc points to. LEFT-REMOTE\(_X\) and RIGHT-REMOTE\(_X\) are similar to LEFT-EDGE\(_X\) and RIGHT-EDGE\(_X\), yet these two transitions create remote edges, creating reentrancies. The X-labelled edge will be assigned a Remote attribute.

Swap deals with non-planar graphs (a generalisation of non-projective trees), in other words, discontinuous constituents. It pops the second node on \(S\) and adds it to the top of \(B\). Finish is the terminal transition, which pops the Root node and marks the transition state as terminal.

In IMPLICIT-EAGER, we introduce a new transition IMPLICIT\(_X\) adding an implicit node to the buffer and attaching it with a labelled edge in one step. In IMPLICIT-STANDARD, we simplify the existing NODE transition to only create a node without attaching it, with the purpose of treating implicit units like primary ones and generating them dynamically. We elaborate their designs in Section 4.1 and Section 4.2. Table 3 shows the transition set.

4.1 IMPLICIT-EAGER

Besides the nine transitions described above, IMPLICIT-EAGER introduces the IMPLICIT\(_X\) transition, which creates a new unit on \(B\) as the child of \(s_0\), with an X-labelled edge. The IMPLICIT action is different from the NODE action of IMPLICIT-STANDARD in the sense that the integrally generated edge makes the new node a child of \(s_0\) rather than its parent, as in NODE\(_X\). Equally importantly, the new node is prohibited to have any child in contrast to the primary nodes that the NODE\(_X\) action generates.

4.2 IMPLICIT-STANDARD

IMPLICIT-STANDARD adopts a more modular approach. Rather than complicating the transition systems, it treats primary non-terminal nodes and implicit nodes equally by simplifying the NODE\(_X\) action, making it generate a new unit on the buffer without attaching it with any (labelled) edge. We assume primary non-terminal nodes and implicit nodes are identical in essence, thus handling them without discrimination. Whenever an ungenerated child or parent of \(s_0\) is found, NODE is executed so that a concept node will be created on \(B\). This action does not cope with edge generation; the work is left to LEFT-EDGE\(_X\) or RIGHT-EDGE\(_X\). In the oracle, we can tell whether the node is primary or implicit by observing its relations. If the newly created node is the child of \(s_0\) and does not have any descendants, it is an implicit node; otherwise, a primary node.

5 Experiments

5.1 Data Preprocessing

We convert UCCA XML data to MRP format using the open-source mtool software.\(^3\) As the UCCA data provided in MRP 2019 shared task did not contain implicit information, HIT-SCIR 2019 is not designed to read this information in our dataset. We modify the parser to read node properties, and to

\(^3\)https://github.com/cfmrp/mtool
Table 4: Statistics of train, dev and evaluation set in Original EWT and Revisited Implicit EWT. For each set, number of sentences, number of tokens, number of nodes, number of instances of 6 implicit categories and their sum are listed.

<table>
<thead>
<tr>
<th>Data</th>
<th># Sentences</th>
<th># Tokens</th>
<th># Nodes</th>
<th># Edges</th>
<th># Deictic</th>
<th># Generic</th>
<th># Genre-based</th>
<th># Type-i</th>
<th># Non-s</th>
<th># Iterated-s</th>
<th># Implicit sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWT Train</td>
<td>2723</td>
<td>44751</td>
<td>59654</td>
<td>95761</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>151</td>
</tr>
<tr>
<td>EWT Dev</td>
<td>554</td>
<td>5394</td>
<td>7534</td>
<td>11987</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>56</td>
</tr>
<tr>
<td>EWT Eval</td>
<td>538</td>
<td>5381</td>
<td>7431</td>
<td>11907</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>56</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td>not refined</td>
<td></td>
</tr>
<tr>
<td>IMP Train</td>
<td>285</td>
<td>2671</td>
<td>3936</td>
<td>6146</td>
<td>87</td>
<td>59</td>
<td>103</td>
<td>3</td>
<td>18</td>
<td>4</td>
<td>274</td>
</tr>
<tr>
<td>IMP Dev</td>
<td>59</td>
<td>540</td>
<td>781</td>
<td>1217</td>
<td>11</td>
<td>15</td>
<td>19</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>IMP Eval</td>
<td>49</td>
<td>489</td>
<td>709</td>
<td>1106</td>
<td>9</td>
<td>12</td>
<td>25</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>61</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>107</td>
<td>86</td>
<td>147</td>
<td>6</td>
<td>36</td>
<td>9</td>
<td>391</td>
</tr>
</tbody>
</table>

Table 5: Experiment results on Revisited Implicit EWT in percents. For primary edges, remote edges, and implicit prediction, listed are Labelled Precision (LP), Labelled Recall (LR) and Labelled F-score (LF). In addition, Unlabelled precision (UL), Unlabelled Recall (UR) and Unlabelled F-score are also listed for implicit evaluation.

<table>
<thead>
<tr>
<th></th>
<th>Primary</th>
<th></th>
<th></th>
<th></th>
<th>Remote</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Implicit</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LP</td>
<td>LR</td>
<td>LF</td>
<td></td>
<td>LP</td>
<td>LR</td>
<td>LF</td>
<td></td>
<td></td>
<td>Implicit</td>
<td>UP</td>
<td>UR</td>
<td>UF</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.495</td>
<td>0.467</td>
<td>0.480</td>
<td>0.538</td>
<td>0.304</td>
<td>0.339</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IMPLICIT-EAGER</td>
<td>0.503</td>
<td>0.472</td>
<td>0.487</td>
<td>0.333</td>
<td>0.100</td>
<td>0.134</td>
<td>(7/21)</td>
<td>(7/50)</td>
<td>(7/21)</td>
<td>(7/50)</td>
<td>0.354</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMPLICIT-STANDARD</td>
<td>0.474</td>
<td>0.431</td>
<td>0.451</td>
<td>0.438</td>
<td>0.280</td>
<td>0.341</td>
<td>(9/22)</td>
<td>(9/50)</td>
<td>(9/22)</td>
<td>(9/50)</td>
<td>0.306</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Experimental Setup

Our parsers use stack LSTM to stabilize gradient descent process and speed up training; we enrich contextual information by employing the pre-trained language model BERT as a feature input (Graves, 2013; Devlin et al., 2018). The model is implemented in AllenNLP (Gardner et al., 2018). We use the HIT-SCIR 2019 parser as the baseline for comparison. We keep the same hyperparameters as Che et al. (2019) except batch size, adjusted from 8 to 4 due to resource constraints. We do not tune hyperparameters on either the original or revisited dataset.\(^5\)

We use the train, validation and evaluation split from Hershcovich et al. (2019b), which was originally from UD EWT, with the ratio of 0.75, 0.125 and 0.125. The evaluation set has been validated as the gold standard. Table 4 shows detailed statistics of train, dev and eval set of both the original and revisited dataset, on which we trained the baseline parser, IMPLICIT-EAGER and IMPLICIT-STANDARD.\(^6\)

6 Results

Table 5 presents experimental results on Revisited Implicit EWT by three parsers, the baseline HIT-SCIR 2019 parser, IMPLICIT-EAGER and IMPLICIT-STANDARD on Revisited Implicit EWT. Regarding performance on the dataset, the baseline is not able to predict implicit argument as expected. However, both IMPLICIT-EAGER and IMPLICIT-STANDARD managed to predict implicit arguments.

Based on the evaluation method mentioned in section 3.1, IMPLICIT-EAGER’s labelled precision and labelled recall on Revisited Implicit EWT are 0.333 and 0.14; the unlabelled precision and unlabelled recall are 0.428 and 0.18. For the primary edge and remote edge evaluation, noticeably, IMPLICIT-EAGER also outperforms the baseline on primary edges by 0.007 in F-score on the revisited dataset.

Even though IMPLICIT-STANDARD has the worse results in terms of primary parsing, it gains boosted performance on all targets in implicit evaluation. Its unlabelled implicit precision, recall and F-score are 0.5, 0.22 and 0.306, defeating IMPLICIT-EAGER by 0.072, 0.04 and 0.052, respectively.

For labelled evaluation, IMPLICIT-STANDARD surpasses IMPLICIT-EAGER even further; the labelled precision, recall and F-score are 0.419, 0.180 and 0.250, respectively exceeding IMPLICIT-EAGER by 0.076, 0.04 and 0.053.

Table 6 presents the three parsers’ performances on Original EWT. The baseline produced better results on primary edges and remote edges on Original EWT.

The reason why IMPLICIT-STANDARD outper-
forms IMPLICIT-EAGER on implicit evaluation but decrease in accuracy on primary evaluation might be attributed to its equal treatment of primary nodes and implicit nodes.

7 Discussion

As is indicated in Table 5, IMPLICIT-EAGER and IMPLICIT-STANDARD successfully predicted respectively seven and nine implicit arguments with the correct fine-grained implicit labels. In the unlabelled evaluation, nine and 11 implicit arguments were predicted each.

Table 7 shows the confusion matrix of the performances of IMPLICIT-EAGER and IMPLICIT-STANDARD on the evaluation set of Revisited Implicit EWT. Both parsers have predicted roughly the same amount of implicit arguments, 22 and 21, respectively.

Noticeably, IMPLICIT-EAGER has emitted 12 Deictic implicit arguments, accounting for 57.1% of all predictions, 14 and 66.7% if including partial matches as Deictic & Generic and Deictic & Genre-based. While IMPLICIT-STANDARD has a more uniform distribution over prediction categories. Deictic, Generic and Genre-based are the most predicted ones. Moreover, one Non-specific implicit argument, despite being wrongly labelled, is also predicted by IMPLICIT-STANDARD, while IMPLICIT-EAGER has never predicted labels other than Deictic, Generic and Genre-based, which have significantly more instances in the training set.

Both implicit parsers have predicted correctly four Deictic and three Generic & Genre-based implicit arguments. Besides, IMPLICIT-STANDARD managed to predict two more correct Genre-based while IMPLICIT-EAGER has never predicted successfully stand-alone Genre-based implicit argument. IMPLICIT-STANDARD has a remarkably higher labelled precision of 66.7% in Deictic than IMPLICIT-EAGER of 33.3%, while the latter has a higher precision of 75% in Generic & Genre-based than IMPLICIT-STANDARD of 50%. Lamentably, neither of the implicit parsers has emitted prediction for Type-identifiable nor Iterated-set. It is necessarily expected as both categories have less than five instances in the training set.

Although this paper focuses on fine-grained implicit Participants, there are already some implicit arguments of other categories annotated in the foundational layer, especially Process and Center. Interestingly, in the sentence “Fresh and excellent quality,” IMPLICIT-STANDARD generated two implicit arguments, Genre-based and Process in the Scene “Fresh.” It means not only that it infers the Scene misses a Genre-based Participant, but also a Process, i.e., the main relation that evolves in time.

8 Related Work

Parsing implicit argument was introduced into NLP by Ruppenhofer et al. (2009); Gerber and Chai (2010, 2012), but has been coarse-grained and annotated within Nombank (Meyers et al., 2004), limited to ten nominal predicates.

Bender et al. (2011) identified ten relevant linguistic phenomena, ran several parsers and associated their output with target dependencies. Roth and Frank (2015) used a rule-based method for identifying implicit arguments, which depends on semantic role labelling and coreference resolution. Similarly, Silberer and Frank (2012); Chiarcos and
Schenk (2015); Schenk and Chiarcos (2016) proposed parsing methods for SemEval 2010 data, but they are only able to parse implicit arguments on a coarse level as well. Cheng and Erk (2018) built a narrative cloze model and evaluated it on Gerber and Chai (2010)’s dataset.

9 Conclusion

Implicit arguments are pervasive in the text but have not been well studied from a general perspective in NLP. In this work, we revisited a recently proposed fine-grained implicit argument typology by addressing its current deficiencies. We annotated a corpus based on the revised guidelines and designed an evaluation metric for measuring implicit argument parsing performance, demonstrating the annotation’s reliability with a superior inter-annotator agreement comparing to other fine-grained implicit argument studies. The dataset will be available to facilitate relevant research.

We introduced the first semantic parser, with two different transition systems, that can handle and predict implicit nodes dynamically, and label them with promising accuracy as part of the meaning representations. We evaluated it on the new dataset and found that some types of implicit arguments are harder to parse than others and that a simpler transition system performs better on parsing implicit arguments at the cost of primary parsing. The fine-grained implicit argument task is challenging and calls for further research.

In future work, we plan to create a large resource of implicit arguments by automatically extracting them from various linguistic constructions in unlabelled text, use it for pre-training of language models, and evaluate them on our and other datasets to gain more insights into this linguistic phenomenon. A post-processing baseline to find implicit arguments after parsing the whole graph would also be interesting for future investigation.

10 Acknowledgments

The authors thank Miryam de Lhoneux and the anonymous reviewers for their helpful feedback.

References


Appendices

A Inter-annotator Confusion Matrix

Table 8 shows the confusion matrix for measuring inter-annotator agreement on the evaluation set. The unlabelled F-score is 91.3%, and the labelled F1-score is 73.8%. The Cohen’s κ between two annotators is 69.3%.

<table>
<thead>
<tr>
<th></th>
<th>UNMATCHED</th>
<th>Nonspecific</th>
<th>Nonspecific</th>
<th>Generic</th>
<th>Nonspecific</th>
<th>Type-identifiable</th>
<th>Deictic</th>
<th>Deictic</th>
<th>Iterated-set</th>
<th>Generic</th>
<th>Generic</th>
<th>Genre-based</th>
<th>Iterated/set</th>
<th>Type-identifiable</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonspecific</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Nonspecific</td>
<td>Deictic</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Nonspecific</td>
<td>Genre-based</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Nonspecific</td>
<td>Type-identifiable</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Genre-based</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Generic</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Iterated-set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Type-identifiable</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Generic</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Iterated-set</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Type-identifiable</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Deictic</td>
<td>Generic</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

B Hyperparameter Settings

Our parsers use stack LSTM to stabilize gradient descent process and speed up training; we enrich contextual information by employing the pre-trained language model BERT as a feature input. We keep the same hyperparameter setting for all three parsers, the baseline parser HIT-SCIR 2019, IMPLICIT-EAGER and IMPLICIT-STANDARD. The setting is shown as the Table 9.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden dimension</td>
<td>20</td>
</tr>
<tr>
<td>Action dimension</td>
<td>50</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>$\beta_1, \beta_2$</td>
<td>0.9, 0.99</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Layer dropout</td>
<td>0.2</td>
</tr>
<tr>
<td>Recurrent dropout</td>
<td>0.2</td>
</tr>
<tr>
<td>Input dropout</td>
<td>0.2</td>
</tr>
<tr>
<td>Batch size</td>
<td>4</td>
</tr>
<tr>
<td>Epochs</td>
<td>50</td>
</tr>
<tr>
<td>Base learning rate</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>BERT learning rate</td>
<td>$5 \times 10^{-5}$</td>
</tr>
<tr>
<td>Gradient clipping</td>
<td>5.0</td>
</tr>
<tr>
<td>Gradient norm</td>
<td>5.0</td>
</tr>
<tr>
<td>Learning rate scheduler</td>
<td>slanted triangular</td>
</tr>
<tr>
<td>Gradual Unfreezing</td>
<td>True</td>
</tr>
<tr>
<td>Cut Frac</td>
<td>0.1</td>
</tr>
<tr>
<td>Ratio</td>
<td>32</td>
</tr>
</tbody>
</table>

C Training Details

As Table 10 shows, the training time is 2 days 22 hours for the baseline on Original UCCA EWT (50 epochs). Best epoch is 3rd; 3 hours for the baseline on Revisited Implicit EWT (30 epochs). Best epoch is 22nd; 1 day 8 hours and 1 day 19 hours for IMPLICIT-EAGER and IMPLICIT-STANDARD on Original UCCA EWT (10 epochs, 13 epochs), respectively, with the best epoch being the 3rd and 2nd; And finally, 6 hours and 8 hours for IMPLICIT-EAGER and IMPLICIT-STANDARD on Revisited Implicit EWT (50
epochs). The best epoch is 21st and 37th, respectively. One can see that all parsers achieved the best performance at an early stage on Original UCCA EWT. However, both implicit parsers took longer time to train on Original UCCA EWT than the baseline.

<table>
<thead>
<tr>
<th></th>
<th>Original Full UCCA EWT</th>
<th>Revisited Implicit Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training time</td>
<td># Epochs</td>
</tr>
<tr>
<td>Baseline</td>
<td>2 days 22 h</td>
<td>50</td>
</tr>
<tr>
<td>IMPLICIT-EAGER</td>
<td>1 day 8 h</td>
<td>10</td>
</tr>
<tr>
<td>IMPLICIT-STANDARD</td>
<td>1 day 19 h</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 10: Training details of the baseline, IMPLICIT-EAGER and IMPLICIT-STANDARD on original UCCA EWT and Revisited Implicit EWT, including training times, the number of best epoch and total epochs.