Semi-automatic Translations of Data Privacy Policies into Controlled NaturalLanguages

IRFAN KHAN TANOLI[1], YASEEN KHAN TANOLI[2] AND ASIF KHALID QURESHI[3]

ABSTRACT: **Natural languages (NLs) are a simple and understandable form for presenting knowledge. However, they are ambiguous and it turns out to be quite complex to process them with machines. Controlled Natural Languages (CNLs) are usually simpler versions of NLs that are obtained by restricting the grammar and vocabulary to reduce, or even eliminate, ambiguity and complexity. CNLs look informal like NLs and are easy to read and understand and can be easily be transformed into machine-readable forms. In this paper, we present NLPT for semi-automatic translation of privacy statements, from NL to a controlled natural one, to improve machine processing. To assess the performance, we experiment on a large set of Twitter data policies. Here, we consider two main aspects i) the translation of social network data privacy policy and ii) the efficiency and efficacy of the proposed system. We also perform an empirical analysis of the results and conclude that our system can be used to translate input policies effectively and efficiently.**

INDEX TERMS: **Controlled Natural Languages, Data Modeling, Natural Language Processing, Social Networks Policies.**

I. INTRODUCTION

Social Networks (SNs) have a great impact on our everyday life. Users increasingly rely on SNs to share their opinions, plan activities, exchange information, and establish social relationships [1]. SNs interactions usually require the exchange of user's data for a variety of purposes, including the provisioning of services. The collection, usage, and sharing of user's data are usually regulated by SNs (e.g., Facebook data [2], Twitter pri- vacy policies [3], Google privacy policies [4]). Usually, publish in English NL, the policies [5] describe the terms and conditions under which the provider will manage the data in terms of e.g., authorised, obliged, or denied. Although the use of the English language en- ables end-users to read and understand the operations allowed (or obliged, or denied) on their data, a key issue relies on the fact that a plain NL cannot be used as the input language for a policy-based software infras- tructure devoted to automatic policy management [6].

Previously in [6], we presented an approach to trans- late NL privacy policies, as they appear on social net-works websites, into a CNL, namely CNL4DSA *Con- trolled Natural Language for Data Sharing Agreements* [7]. To assess the validity of our translator, we run the NLPT system on a small set of Facebook data policies. Initially, NLPT can translate only a few sets of data polices [6]. In this paper, we present the NLPT 2.0 with vast improvements from the previous version, equipped with a user-friendly GUI where non-expert users can input policies in NL sentences for transla- tion purposes. Apart, the system is designed to assist researchers to evaluate policies specification of data privacy in social networks, also for other application domains (e.g., e-health, e-commerce, etc).

The rest of paper is structured as follows: Section II describes the overall system architecture. Section IV presents experimental setup. Section V discusses the re- sults. The final section VI outlines directions for future work and draws the conclusions.

[1]Gran Sasso Science Institute, L'Aquila 67100, Italy (e-mail: irfankhan.tanoli@gssi.it)
[2]University of Beira Interior, Covilhã 6201-001, Portugal (e-mail: yaseen.khan.tanoli@ubi.pt)
[3]Sir Syed University of Engineering and Technology, Karachi City, Sindh, Pakistan (e-mail: asif.khalid@ssuet.edu.pk)

Corresponding author: Irfan Khan Tanoli (e-mail: irfan.khan.tanoli@ubi.pt).

II. ARCHITECTURE AND GRAPHICAL USER INTERFACE

In [6], we present the methodological approach including all necessary steps required for policy translation. We have illustrated our approach on Facebook Policies translations. In this section, we show the next prototypical version of the system so-called 'Natural Language Policy Translator' (NLPT) 2.0. The system enables generic social network NL data policies translation into CNL4DSA. Here, we highlight the overall architecture of NLPT 2.0, including its Graphical User Interface (GUI). In particular, we introduce the role of the Policy Writer, an end-user to input the original policy following the instructions available in the usermanual.

A. ARCHITECTURE OF NLPT

Natural Language Policy Translator (NLPT) 2.0 is composed of different components. Figure 1 reports the NLPT high level architecture. The details of each com- ponent are as follows: Policy Writer is an entity that

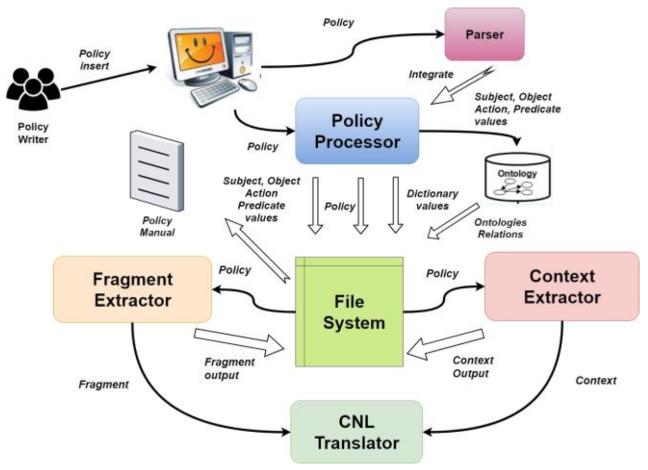


FIGURE 1: System Architecture NLPT 2.0

appropriately writes the policy. The major need for the presence of a Policy Writer is to avoid the possible incorrect tagging operated by the dependency parser. There are different styles of writing the policies, accord- ing to the different online service providers: if a privacy statement is long and complex, quite naturally it contains several different predicates. In this case, it may happen that the parser does not parse the statement properly, or it wrongly tags few terms. This issue has a direct impact on the policy translation, particularly for the extraction of actions and predicates.

As an example, let the reader consider the following policy: (https://twitter.com/en/privacy)

M1: 'We also collect contact information if you choose to upload, sync, or import it from a device'.

Processing M1 using the Dependency parser, we obtain the output shown in Figure 2.

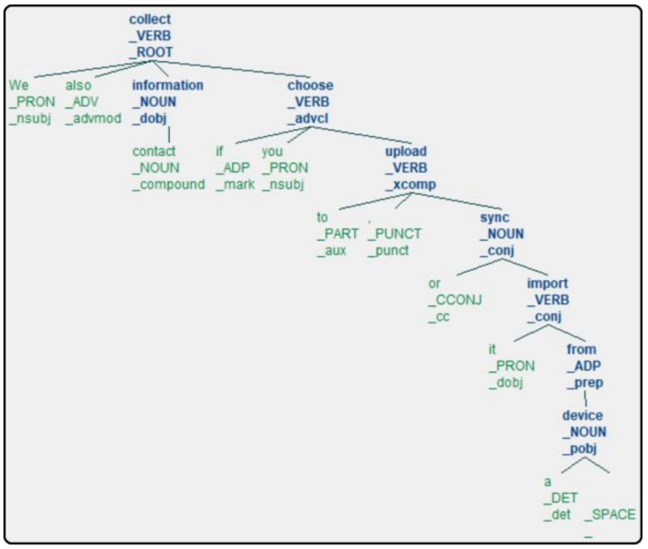


FIGURE 2: M1 upon dependency parser processing-1

The Policy Parser (PP) labels 'sync' as a noun, while this should be tagged as a verb. However, if we simply edit the policy to specify it properly, i.e., with the right combination of Subject Verb and Object (SVO), then the parser works fine. An adequate rephrasing is:

Rephrase M1: 'We also collect data if you choose_to_upload data or you sync data or you import data from a device'.

The resulting output in Figure 3. It is also worth noting that terms like 'information', 'content', 'contact information, 'content and information, and so on, need to be replaced with the generic word 'data', for a more accurate tagging. Also, we introduce (_) in an infinitive form verb i.e., 'choose_to_upload' which makes the translator recognize as a single word.

Therefore, to mitigate such issues, we introduce the role of the Policy Writer, who does not need to be a CNL expert or a domain expert, just an end-user who should know the correct way to write and enter the policy in the system. To train the Policy Writer, a Policy Manual is provided with all the instructions.

The next component Policy Processor ((PR) 1 processes the policy by applying sentence and words tokenization, stop-words removal, and the definition and creation of the dictionary for the uni-gram training for tagging object(s) and subject(s) in the policy. PR also lists the number of subject(s), object(s), action(s) and predicate(s). The Policy Processor is entirely developed with the combination of NLTK [8], and Spacy [9]. The output of this processing is then stored in a file using the Python OS.System function.

The identified subject(s), object(s), action(s) and predicate(s) is/are given in input to the Ontology Builder, already equipped with some predefined pred- icates, e.g., hasRole, hasID, hasOwner, for subjects,

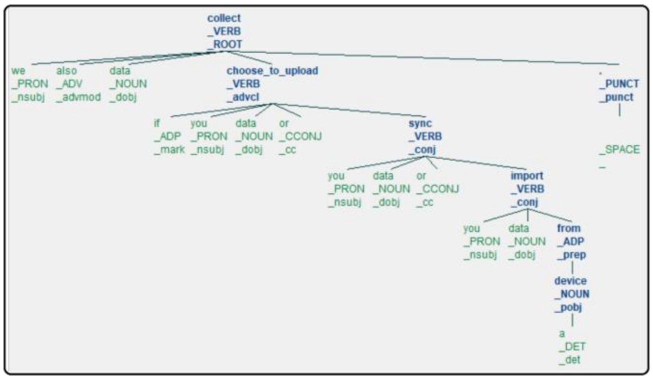


FIGURE 3: Policy upon dependency parser processing-3

hasCategory for objects and object/subject isRelatedto, isPartof object/subject. We also define the classes: sub- ject, object, action, category, predicate, id, owner, and role.

The Ontology Builder is implemented using Owl ontologies [10] and Owl ready [11], a Python module to load them. The ontologies vocabulary is updated manually, whenever any new term is required to process the policy. The result is stored in a file, later used by the Context Extractor (CE).

Next, the Fragment Extractor (FE) extracts the fragment ‘subject action object’. This component is implemented using logic programming for fragment recognition in a policy, i.e., it maps subject, action, and object with the identified values and shows the output. For example, Fragment Extractor efficiently extracts the fragment, i.e., *‘we collect data’: ‘subject action object’* (See Figure 10). The output is again stored in a separate file, saved in the file system, and later used by the CNL Translator (CNLT).

The Context Extractor (CE) extracts the CNL4DSA contexts. This component is developed using logic programming, and it also recalls the output of the On- tology Builder, to retrieve the contexts with predicates hasRole and hasCategory, if any. If we consider M1, the expected context is:

subject hasRole: ‘we’ AND subject hasRole: ‘you’ AND object hasCategory: ‘data’ AND subject (you) predicate (choose_to_upload) object (data) AND subject (you) predicate (sync) object (data) AND subject (you) predicate (import) object (data) is a composite context.

The output is again stored in a separate file for the CNL Translator usage. CNL Translator (CNLT) recalls the outputs of FE and CE from the file-system and shows the policy translation into CNL4DSA. CNLT has been designed to discriminate between authorization, obligation, and prohibition fragments.

The component *Complete Translation into CNL (CTCNL)* orchestrates the previously introduced components, by calling Policy Parser first, following Policy Processor, next Ontology Builder, afterwards Fragment Extractor, followin Context Extractor and, finally CNL Translator. Figure 4 shows the complete process flow of translation as a single iteration.

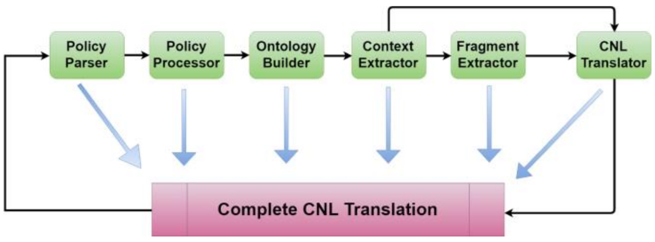


FIGURE 4: Complete CNL Translation Process

III. GRAPHICAL USER INTERFACE

The Graphical User Interface GUI of NLPT is developed with the Python Automatic GUI Generator (PAGE) [12]. Figure III shows the main screen. The right side and bottom side of the window show the components (under the form of buttons). The user inputs the policy in the text area. If the text area field is empty, an empty field error is prompted, Figure 5.

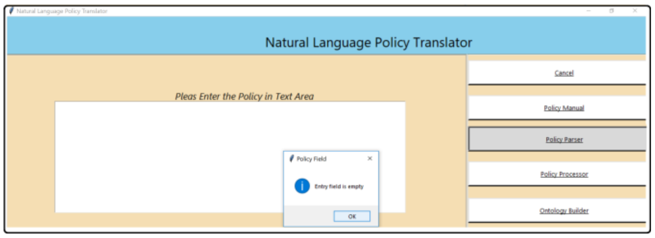

Natural Language Policy Translator
Natural Language Policy Translator
Pleas Enter the Policy in Text Area
Policy Field
Entry field is empty
OK
Cancel
Policy Manual
Policy Parser
Policy Processor
Ontology Builder

FIGURE 5: Empty Field Error

Consider the following example Twitter policy P1:

P1: We collect the content, communications, and other information you provide when you use our Prod- ucts, including when you sign up for an account, create or share content, and message or communicate with others (https://twitter.com/en/privacy).

First, the Policy Writer needs to check whether the policy is accurately parsed by the parser. Otherwise, the Policy Writer should rephrase the policy taking as guide the policy manual:

Rephrase P1: we collect the data that data you provide when you use our products including when you signup for an account, when you create the data, when you share the data when you message with others, and when you communicate with others.

The policy is not entirely rephrased, however, there is the need to add punctuation and remove some parts, to make the constructs of the sentence easily recognizable by the parser. As anticipated above, words like ‘content, communication, other information are rephrased into the generic word ‘data’.

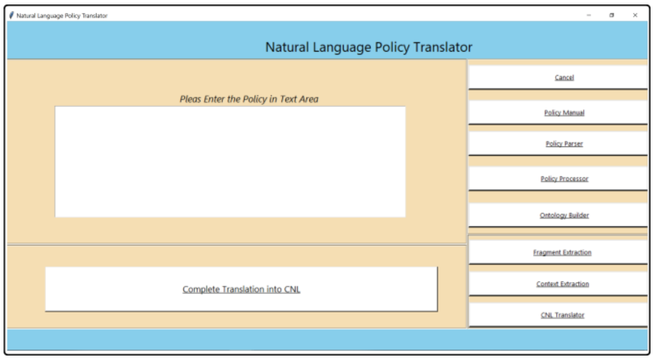
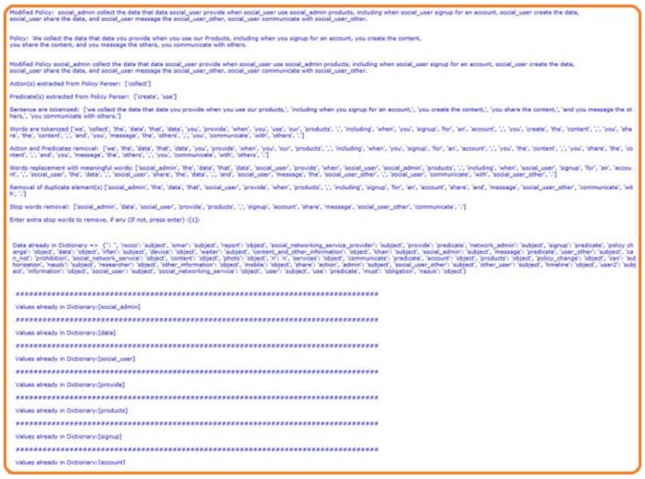

Natural Language Policy Translator
Natural Language Policy Translator
Pleas Enter the Policy in Text Area
Complete Translation into CNL
Cancel
Policy Manual
Policy Parser
Policy Processor
Ontology Builder
Fragment Extraction
Context Extraction
CNL Translator

FIGURE 6: NLPT Graphical User Interface

When a policy is enter in NLPT, Figure 7 and 8 show the outputs of Policy Processor. The output of Ontology Builder, i.e., ontologies with the values of subject and object is shown in Figure 9. The output generated using Fragment and Context extractor show in Figure 10.



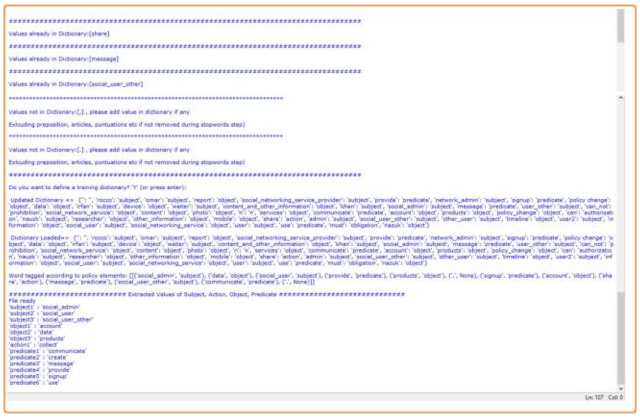
```
Modified Policy:  social_admin collect the data that data social_user provide when social_user use social_admin products, including when social_user signup for an account, social_user create the data,
social_user share the data, and social_user message the social_user_other, social_user communicate with social_user_other.

Policy:  We collect the data that data you provide when you use our Products, including when you signup for an account, you create the content,
you share the content, and you message the others, you communicate with others.

Modified Policy social_admin collect the data that data social_user provide when social_user use social_admin products, including when social_user signup for an account, social_user create the data,
social_user share the data, and social_user message the social_user_other, social_user communicate with social_user_other.

Action(s) extracted from Policy Parser:  ['collect']

Predicate(s) extracted from Policy Parser:  ['create', 'use']

Sentence are tokenized:  ['we collect the data that data you provide when you use our products,', 'including when you signup for an account,', 'you create the content,', 'you share the content,', 'and you message the ot
hers,', 'you communicate with others.']

Words are tokenized ['we', 'collect', 'the', 'data', 'that', 'data', 'you', 'provide', 'when', 'you', 'use', 'our', 'products', ',', 'including', 'when', 'you', 'signup', 'for', 'an', 'account', ',', 'you', 'create', 'the', 'content', ',', 'you', 'sha
re', 'the', 'content', ',', 'and', 'you', 'message', 'the', 'others', ',', 'you', 'communicate', 'with', 'others', '.']

Action and Predicates removal:  ['we', 'the', 'data', 'that', 'data', 'you', 'provide', 'when', 'you', 'our', 'products', ',', 'including', 'when', 'you', 'signup', 'for', 'an', 'account', ',', 'you', 'the', 'content', ',', 'you', 'share', 'the', 'co
ntent', ',', 'and', 'you', 'message', 'the', 'others', ',', 'you', 'communicate', 'with', 'others', '.']

Words replacement with meaningful words: ['social_admin', 'the', 'data', 'that', 'data', 'social_user', 'provide', 'when', 'social_user', 'social_admin', 'products', ',', 'including', 'when', 'social_user', 'signup', 'for', 'an', 'accou
nt', ',', 'social_user', 'the', 'data', ',', 'social_user', 'share', 'the', 'data', ',', 'and', 'social_user', 'message', 'the', 'social_user_other', ',', 'social_user', 'communicate', 'with', 'social_user_other', '.']

Removal of duplicate element(s) ['social_admin', 'the', 'data', 'that', 'social_user', 'provide', 'when', 'products', ',', 'including', 'signup', 'for', 'an', 'account', 'share', 'and', 'message', 'social_user_other', 'communicate', 'wit
h', '.']

Stop words removal:  ['social_admin', 'data', 'social_user', 'provide', 'products', ',', 'signup', 'account', 'share', 'message', 'social_user_other', 'communicate', '.']

Enter extra stop words to remove, if any (If not, press enter) :[1]:


 Data already in Dictionary =>  {'': '', 'rocco': 'subject', 'omar': 'subject', 'report': 'object', 'social_networking_service_provider': 'subject', 'provide': 'predicate', 'network_admin': 'subject', 'signup': 'predicate', 'policy ch
 ange': 'object', 'data': 'object', 'irfan': 'subject', 'device': 'object', 'waiter': 'subject', 'content_and_other_information': 'object', 'khan': 'subject', 'social_admin': 'subject', 'message': 'predicate', 'user_other': 'subject', 'ca
 n_not': 'prohibition', 'social_network_service': 'object', 'content': 'object', 'photo': 'object', 'n': 'n', 'services': 'object', 'communicate': 'predicate', 'account': 'object', 'products': 'object', 'policy_change': 'object', 'can': 'aut
 horization', 'nauzk': 'subject', 'researcher': 'object', 'other_information': 'object', 'mobile': 'object', 'share': 'action', 'admin': 'subject', 'social_user_other': 'subject', 'other_user': 'subject', 'timeline': 'object', 'user2': 'subj
 ect', 'information': 'object', 'social_user': 'subject', 'social_networking_service': 'object', 'user': 'subject', 'use': 'predicate', 'must': 'obligation', 'nazuk': 'object'}


  ################################################################################

  Values already in Dictionary:[social_admin]

  ################################################################################

  Values already in Dictionary:[data]

  ################################################################################

  Values already in Dictionary:[social_user]

  ################################################################################

  Values already in Dictionary:[provide]

  ################################################################################

  Values already in Dictionary:[products]

  ################################################################################

  Values already in Dictionary:[signup]

  ################################################################################

  Values already in Dictionary:[account]
```

FIGURE 7: Output of Policy Processor.a

The CNLT displays the input policy and the CNL4DSA output, hiding the internal process. It also discriminates between authorizations, prohibi- tions, and obligations. Referring to P1 and its output in Figure 11 shows the result. Moreover, it is also possible to view the outputs of each component altogether by using Complete Translation button and the order of op- eration is as follows: Policy Parser -> Policy Processor -> Ontology Builder -> CE -> FE -> CNLT as presented in Figure 4.

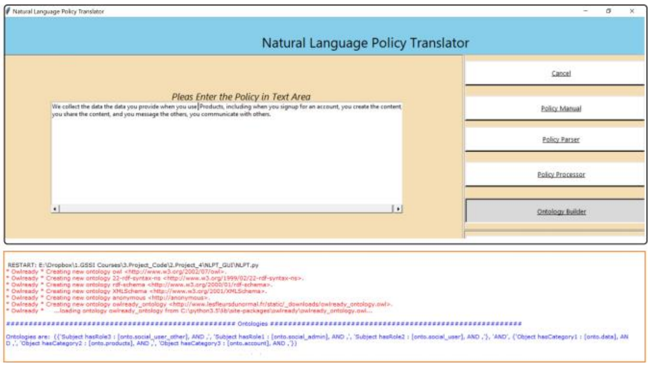
IV. EXPERIMENTAL SETUP

In this section, we run the experiments on a quantita- tive basis, to assess the *NLPT* performances. We select a subset of Twitter data policies [3]. The main goal of



```
################################################################################

Values already in Dictionary:[share]

################################################################################

Values already in Dictionary:[message]

################################################################################

Values already in Dictionary:[social_user_other]

********************************************************************************

Values not in Dictionary:[,] , please add value in dictionary if any

Excluding preposition, articles, puntuations etc if not removed during stopwords step)

********************************************************************************

Values not in Dictionary:[.] , please add value in dictionary if any

Excluding preposition, articles, puntuations etc if not removed during stopwords step)

################################################################################

Do you want to define a training dictionary? 'Y' (or press enter):

 Updated Dictionary =>  {'': '', 'rocco': 'subject', 'omar': 'subject', 'report': 'object', 'social_networking_service_provider': 'subject', 'provide': 'predicate', 'network_admin': 'subject', 'signup': 'predicate', 'policy change':
'object', 'data': 'object', 'irfan': 'subject', 'device': 'object', 'waiter': 'subject', 'content_and_other_information': 'object', 'khan': 'subject', 'social_admin': 'subject', 'message': 'predicate', 'user_other': 'subject', 'can_not':
'prohibition', 'social_network_service': 'object', 'content': 'object', 'photo': 'object', 'n': 'n', 'services': 'object', 'communicate': 'predicate', 'account': 'object', 'products': 'object', 'policy_change': 'object', 'can': 'authorizati
on', 'nauzk': 'subject', 'researcher': 'object', 'other_information': 'object', 'mobile': 'object', 'share': 'action', 'admin': 'subject', 'social_user_other': 'subject', 'other_user': 'subject', 'timeline': 'object', 'user2': 'subject', 'in
formation': 'object', 'social_user': 'subject', 'social_networking_service': 'object', 'user': 'subject', 'use': 'predicate', 'must': 'obligation', 'nazuk': 'object'}

 Dictionary Loaded=>  {'': '', 'rocco': 'subject', 'omar': 'subject', 'report': 'object', 'social_networking_service_provider': 'subject', 'provide': 'predicate', 'network_admin': 'subject', 'signup': 'predicate', 'policy change': 'o
bject', 'data': 'object', 'irfan': 'subject', 'device': 'object', 'waiter': 'subject', 'content_and_other_information': 'object', 'khan': 'subject', 'social_admin': 'subject', 'message': 'predicate', 'user_other': 'subject', 'can_not': 'p
rohibition', 'social_network_service': 'object', 'content': 'object', 'photo': 'object', 'n': 'n', 'services': 'object', 'communicate': 'predicate', 'account': 'object', 'products': 'object', 'policy_change': 'object', 'can': 'authorizatio
n', 'nauzk': 'subject', 'researcher': 'object', 'other_information': 'object', 'mobile': 'object', 'share': 'action', 'admin': 'subject', 'social_user_other': 'subject', 'other_user': 'subject', 'timeline': 'object', 'user2': 'subject', 'inf
ormation': 'object', 'social_user': 'subject', 'social_networking_service': 'object', 'user': 'subject', 'use': 'predicate', 'must': 'obligation', 'nazuk': 'object'}

Word tagged according to policy elements: [[('social_admin', 'subject'), ('data', 'object'), ('social_user', 'subject'), ('provide', 'predicate'), ('products', 'object'), (',', None), ('signup', 'predicate'), ('account', 'object'), ('sha
re', 'action'), ('message', 'predicate'), ('social_user_other', 'subject'), ('communicate', 'predicate'), ('.', None)]]

########################## Extracted Values of Subject, Action, Object, Predicate ##############################
File ready
'subject1' : 'social_admin'
'subject2' : 'social_user'
'subject3' : 'social_user_other'
'object1' : 'account'
'object2' : 'data'
'object3' : 'products'
'action1' : 'collect'
'predicate1' : 'communicate'
'predicate2' : 'create'
'predicate3' : 'message'
'predicate4' : 'provide'
'predicate5' : 'signup'
'predicate6' : 'use'
                                                                                                   Ln: 107   Col: 0
```

FIGURE 8: Output of Policy Processor.b


Natural Language Policy Translator
Natural Language Policy Translator
Pleas Enter the Policy in Text Area
Cancel
Policy Manual
Policy Parser
Policy Processor
Ontology Builder

We collect the data the data you provide when you use Products, including when you signup for an account, you create the content,
you share the content, and you message the others, you communicate with others.

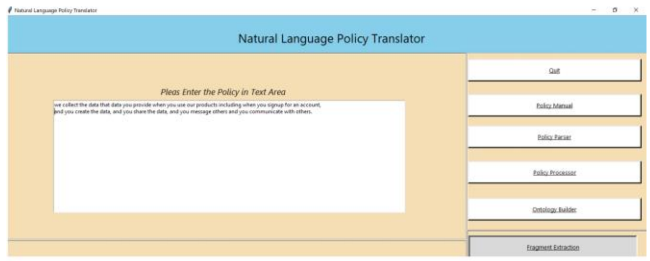
```
RESTART: E:\Dropbox\1.GSSI Courses\3.Project_Code\2.Project_4\NLPT_GUI\NLPT.py
* Owlready * Creating new ontology owl <http://www.w3.org/2002/07/owl>.
* Owlready * Creating new ontology 22-rdf-syntax-ns <http://www.w3.org/1999/02/22-rdf-syntax-ns>.
* Owlready * Creating new ontology rdf-schema <http://www.w3.org/2000/01/rdf-schema>.
* Owlready * Creating new ontology XMLSchema <http://www.w3.org/2001/XMLSchema>.
* Owlready * Creating new ontology anonymous <http://anonymous>.
* Owlready * Creating new ontology owlready_ontology <http://www.lesfleursdunormal.fr/static/_downloads/owlready_ontology.owl>.
* Owlready *     ...loading ontology owlready_ontology from C:\python3.5\lib\site-packages\owlready\owlready_ontology.owl...

################################################### Ontologies ############################################################

Ontologies are:  ({'Subject hasRole3 : [onto.social_user_other], AND ,', 'Subject hasRole1 : [onto.social_admin], AND ,', 'Subject hasRole2 : [onto.social_user], AND ,'}, 'AND', {'Object hasCategory1 : [onto.data], AN
D ,', 'Object hasCategory2 : [onto.products], AND ,', 'Object hasCategory3 : [onto.account], AND ,'})
```

FIGURE 9: Output of Ontology Builder

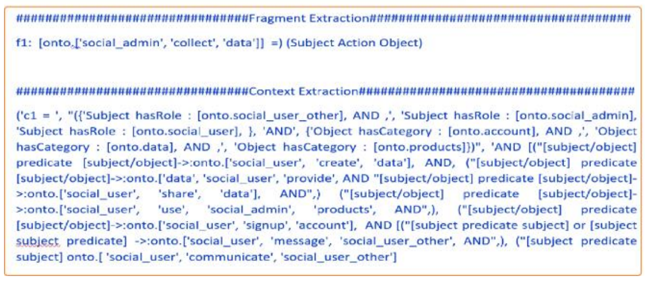
the experiment is to assess whether *NLPT* is capable of translating any social network data policies. To analyze the performances of the different components, we de- fine some evaluation criteria:

Evaluation Criteria:

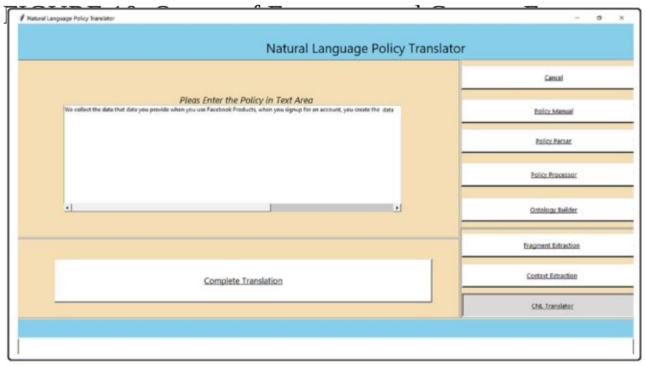
1) How many policies are accurately parsed by Policy Parser (PP), correctly labeling action(s) and predi- cate(s) in a policy?
2) How many times is there the need of updating the dictionary, in terms of subject(s) and object(s), while processing the policy with Policy Processor (PR), until no update is needed?
3) How many times is there the need of updating an existing ontology, or creating a new one until no update is needed?
4) From how many input policies does Fragment Ex- tractor (FE) correctly extract the fragments?
5) From how many input policies does Context Ex- tractor (CE) correctly extract the contexts?
6) From how many input policies is CNLT able to correctly recognize as authorisations, obligations, and prohibitions?
7) What is the Success Rate of PP, FE, CE and CNLT, over the total number of policies?


Natural Language Policy Translator
Natural Language Policy Translator
Pleas Enter the Policy in Text Area
Quit
Policy Manual
Policy Parser
Policy Processor
Ontology Builder
Fragment Extraction

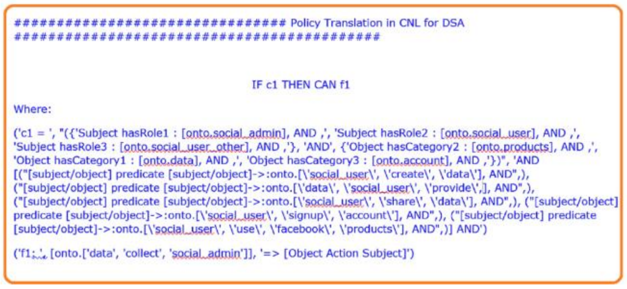
we collect the data that data you provide when you use our products including when you signup for an account,
and you create the data, and you share the data, and you message others and you communicate with others.



```
####################################Fragment Extraction######################################
f1:  [onto.['social_admin', 'collect', 'data']]  =) (Subject Action Object)


####################################Context Extraction#######################################
('c1 = ', "({'Subject hasRole : [onto.social_user_other], AND ,', 'Subject hasRole : [onto.social_admin],
'Subject hasRole : [onto.social_user], }, 'AND', {'Object hasCategory : [onto.account], AND ,', 'Object
hasCategory : [onto.data], AND ,', 'Object hasCategory : [onto.products]})", 'AND [("[subject/object]
predicate [subject/object]->:onto.['social_user', 'create', 'data'], AND, ("[subject/object] predicate
[subject/object]->:onto.['data', 'social_user', 'provide', AND "[subject/object] predicate [subject/object]-
>:onto.['social_user', 'share', 'data'], AND",) ("[subject/object] predicate [subject/object]-
>:onto.['social_user', 'use', 'social_admin', 'products', AND",), ("[subject/object] predicate
[subject/object]->:onto.['social_user', 'signup', 'account'], AND [("[subject predicate subject] or [subject
subject predicate] ->:onto.['social_user', 'message', 'social_user_other', AND",), ("[subject predicate
subject] onto.[ 'social_user', 'communicate', 'social_user_other']
```



We collect the data that data you provide when you use Facebook Products, when you signup for an account, you create the data



```
################################ Policy Translation in CNL for DSA
############################################


                         IF c1 THEN CAN f1

Where:

('c1 = ', "({'Subject hasRole1 : [onto.social_admin], AND ,', 'Subject hasRole2 : [onto.social_user], AND ,',
'Subject hasRole3 : [onto.social_user_other], AND ,'}, 'AND', {'Object hasCategory2 : [onto.products], AND ,',
'Object hasCategory1 : [onto.data], AND ,', 'Object hasCategory3 : [onto.account], AND ,'})", 'AND
[("[subject/object] predicate [subject/object]->:onto.[\'social_user\', \'create\', \'data\'], AND",),
("[subject/object] predicate [subject/object]->:onto.[\'data\', \'social_user\', \'provide\',], AND",),
("[subject/object] predicate [subject/object]->:onto.[\'social_user\', \'share\', \'data\'], AND",), ("[subject/object]
predicate [subject/object]->:onto.[\'social_user\', \'signup\', \'account\'], AND",), ("[subject/object] predicate
[subject/object]->:onto.[\'social_user\', \'use\', \'facebook\', \'products\'], AND",)] AND')

('f1:', [onto.['data', 'collect', 'social_admin']], '=> [Object Action Subject]')
```

FIGURE 11: Output of Fragment and Context Extractor

A. *EXPERIMENTS*

We process Twitter data privacy policies [3] with our prototype system, and the results are evaluated using the evaluation criteria introduced above.

Each policy is processed in each component and the results are noted as follows:

1) Run Policy Parser
   - Count number of policies parsed correctly and parsed incorrectly.
2) Run Policy Processor
   - Count number of policies for which is required to update the vocabulary, with respect to newsubjects, objects and predicates.
3) Run Ontology Builder
   - Count number of policies for which is required to create or update ontologies.
4) Run Fragment Extractor
   - Count number of fragments correctly extracted (in a single iteration and/or in multiple itera- tions)
5) Run Context Extractor
   - Count number of contexts correctly extracted (in a single iteration and/or in multiple iterations)
6) Run CNL Translator
   - Count number of policies for which CNLT suc- cessfully differentiates between authorizations, obligations and prohibitions.

We calculate the success rate of PP, FE, CE, and CNLT, using the following formula:

Success Rate Formula

$$\text{Success Rate} = \frac{X}{T} * 100$$

where:

T = Total number of policies.

X = Number of policies correctly parsed by PP or accurately extracted by FE and CE in a single iteration or; Number of policies correctly recognised as autho- rizations, obligations, and prohibitions by CNLT. Next, we run the NLPT 2.0 on 100 different Twitter data policies and the results are presented and evaluated in section V.

V. RESULTS

The result of each component's performance *WRT* the evaluation criteria described in section IV are shown in Table 1, Table 2, and Table 3 where rows indicate the evaluation criteria, the first column indicates the case study and the other columns provide information about the evaluation. Table 4 summarizes the success rate of PP, FE, CE and CNLT.

A. *OVERALL RESULTS*

For PP, 77 out of 100 policies are correctly parsed by the dependency parser. Initially, for first 100 policies, it is required to update terms for subjects and objects e.g., e.g., 'services', 'account', 'apps', 'you', 'we', 'users' etc. Later, it was not needed as vocabulary updated enough to tag subjects and objects automatically. The same happens for the ontology vocabulary.

The number of correctly extracted fragments is 84. CE correctly extracts 72 policies contexts, for 10 poli- cies, it misses few contexts, for 18 policies, no output displayed. CNLT correctly recognizes 90 policies as authorizations, obligations, and prohibitions.

Following the success rate are calculated as follows:

Case Study	Total Policies	Correct Parsing	Incorrect Parsing	Dictionary Update	Ontology Update
Twitter	100	77	23	18	18

TABLE 1: Policy parser results for Twitter

Case Study	Total Policies	Accurate F.E	Inaccurate F.E	Accurate C.E	Inaccurate C.E
Twitter	100	84	16	72	28

TABLE 2: FE and CE results for Twitter

PP Success Rate = (77/100)*100= 77%

FE Success Rate = (84/100)*100= 84%

CE Success Rate = (72/100)*100= 72%

CNLT Success Rate = (90/100)*100= 90%

The complete results are shown in Table 1, Table 2 and Table 3

The success rate for each component is shown in Ta- ble 4. The PP and FE success rates are above 75%. While CE success rates lay between 55% and 75%. The CNLT performance obtains approximately the 90% success rate. The categorised performance of each component is shown in Table 5.

VI. CONCLUSION

In this paper, we presented the NLPT 2.0 an extended version of our work [6] having new functionalities for translating data privacy policies with their descriptions in NL into phrases of a CNL. The work aims to provide an effective and efficient technique for the formal anal- ysis of privacy policies. Social media sites use natural language (NL) when posting the data policies about the handling of user's data. But these policies are often unclear and ambiguous and as stressed before, are not machine check-able. This is why we resorted to con- trolled natural languages (CNLs). These languages are engineered languages with a strict lexicon, vocabulary, syntaxes, grammatical construction, and pragmatics and are more amenable to machine processing. The target language is for translation is CNL4DSA. This language is closer to a purely natural language and thus eases readability and usability by non-experts.

Case Study	Total Policies	Accurate Distinctionby CNLT	Inaccurate Distinctionby CNLT
Twitter	100	90	10

TABLE 3: Overall results of Twitter

Case Study	P.P Success Rate	F.E Success Rate	C.E Success Rate	CNLT Success Rate
Twitter	77%	84%	72%	90%

TABLE 4: P.P, F.E, C.E and CNLT Success Rates

Case Study	P.P	F.E	C.E	CNLT
Twitter	Excellent	Excellent	Above Average	Excellent

TABLE 5: F.E, C.E and CNLT performance evaluationagainst Success Rate

The experiment on Twitter data policies demon- strated the overall performance of NLPT 2.0 is quite good. Overall results of different system components are encouraging and satisfactory. We identified certain limitations with our system during experimentation, and we are considering improvements in the future. We are investigating the capability of automatically creat- ing/updating the ontologies. We cannot simply input a policy in its original form. The Spacy dependency parser [9] may not be able to correctly parse too com- plex sentences, Quite obviously, an incorrect parsing also has a direct impact on FE and CE performances. We introduced the role of a Policy Writer, whose job is to rephrase the policy following the instructions available in the policy manual.

For now, the whole process is semi-automatic since it requires human intervention at different stages. For future work, we want to experiment with our approach on other social network data policies e.g., Facebook, Google, etc, and also for other domains, e.g., 'e-health privacy policies', e-commerce policies', 'cloud comput- ing privacy policies'. Initially, we only experimented our approach with Twitter policies, but with our un- derstanding, our approach is sufficiently flexible to be used in other domains by just creating/updating the vocabulary in terms of subject and object relevant for the considered social network sites.

Moreover, we have not dealt with policies with more than one fragment. If a verb followed by an infinitive form appears, such as 'choose to upload', we pre- process that form by introducing an underscore ('_'), i.e., 'choose_to_upload', and let the translator consider it as a single word. We do not consider automatic co- reference detection. This is a limitation of the approach and we intend to solve the issue, possibly using existing tools for automatic co-reference detection.

REFERENCES

[1] S. Pais, I. K. Tanoli, M. Albardeiro, and J. Cordeiro, "Unsupervised approach to detect extreme sentiments on social networks," in 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2020, pp. 651–658.

[2] "Data Policy," 2020, URL: https://www.facebook.com/full_data_ use_policy [accessed: 2020-09-15].

[3] "Twitter Privay Policy," 2020, URL: https://twitter.com/en/ privacy [accessed: 2020-09-15].

[4] "Google Privacy and Terms," 2020, URL: https://policies.google. com/privacy [accessed: 2020-09-15].

[5] Cambridge, "Policy," https://dictionary.cambridge.org/ dictionary/english/policy, 2018.

[6] I. K. Tanoli, M. Petrocchi, and R. De Nicola, "Towards automatic translation of social network policies into controlled natural lan-

guage," in 2018 12th International Conference on Research Challenges in Information Science (RCIS). IEEE, 2018.

[7] I. Matteucci, M. Petrocchi, and M. L. Sbodio, "CNL4DSA: a controlled natural language for data sharing agreements," in Sympo- sium on Applied Computing. ACM, 2010, pp. 616–620.

[8] S. Bird, "NLTK: the natural language toolkit," in COLING. Association for Computational Linguistics, 2006, pp. 69–72.

[9] Matthew Honnibal, Ines Montani, "spaCy 101: Everything you need to know," 2017. [Online]. Available: https://www.w3.org/TR/owl2-overview/

[10] W. O. W. Group, "Owl 2 web ontology language," 2012. [Online]. Available: https://www.w3.org/TR/owl2-overview

[11] Owlready, "Owlready documentation," 2018, available at https://pypi.python.org/pypi/Owlready.

[12] D. Rozenberg, "Page - python automatic gui generator," 2018, available at http://page.sourceforge.net/.