Triple evaluation module

version 1.0
July, 2013

Piek Vossen, VU University Amsterdam

piek.vossen@vu.nl

Table of Contents

1. Introduction

Semantic structures derived from natural language can be very complex and represented in many different ways. Consider the following example that is taken from the ECB corpus[1]:

> *Firefighters from multiple agencies responded to Highway 38 near Bryant Street in Mentone about 6:30 p.m..*

The main event in this sentence, expressed by \it{responded}, involves a participant, \it{firefighters from multiple agencies}, a location, \it{Highway 38 near Bryant Street in Mentone}, and a time expression, \it{ 6:30 p.m.}. These elements also have internal structures, such a the fact that the firefighters belong to multiple agencies and that the location is close to Bryant Street and in Mentone. A complete representation of these relations yields a complex structure involving all these relations and uses some formalism. Some of the main relations are represented in the next tuple structure:

```
<tuple id="1" profile="agent-1-sem" profileConfidence="30" sentenceId="s2">
  <event concept="eng-30-00717358-v" confidence="0.703748" lemma="respond" tokens="w34"/>
  <participant lemma="firefighter" role="agent" tokens="w30"/>
  <location lemma="Highway 38" tokens="w36;w37"/>
  <time lemma="6:30 p.m." tokens="w44;w45"/>
</tuple>
```

[1] http://faculty.washington.edu/bejan/data/ECB1.0.tar.gz

In this representation, the element are grounded to tokens of the text and indirectly connected to the event element that governs the other elements. This representation only represents part of the relations in the example.

However, any structure can be decomposed in a set of basic relations between expressions. The Triple module has been designed to represent and evaluate any structure that is expressed by natural language, regardless of the representation of this structure. The structure of a triple is defined by:

1. a relation
2. a range of text tokens that represents the first element: the governing parent expression
3. a range of text tokens that represent the second element: the child expression

The triple elements point back to the tokens of the text. This makes the representation maximally independent of the interpretation of the text. The first element represent the parent element that governs a second element that is a child. In the case of the above structure, the event is the natural element that governs all the other relations. If a structure contains more than two related elements, it can then be defined as a list of two-place relations, each bound to the same governing element. The above structure can thus be represented as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<triples>
<triple id="bus-accident.ont.dep.kaf#1" profile_id="agent-1-sem" profile_confidence="30" relation="agent">
        <elementFirstIds label="event" comment="respond">
                <elementFirst id="w34"/>
        </elementFirstIds>
        <elementSecondIds label="participant" comment="Firefighters">
                <elementSecond id="w30"/>
        </elementSecondIds>
</triple>
<triple id="bus-accident.ont.dep.kaf#1" profile_id="agent-1-sem" profile_confidence="30" relation="location">
        <elementFirstIds label="event" comment="respond">
                <elementFirst id="w34"/>
        </elementFirstIds>
        <elementSecondIds label="location" comment="Highway 38">
                <elementSecond id="w36"/>
                <elementSecond id="w37"/>
        </elementSecondIds>
</triple>
<triple id="bus-accident.ont.dep.kaf#1" profile_id="agent-1-sem" profile_confidence="30" relation="time">
        <elementFirstIds label="event" comment="respond">
                <elementFirst id="w34"/>
        </elementFirstIds>
        <elementSecondIds label="time" comment="6:30 p.m.">
```

```
                <elementSecond id="w44"/>
                <elementSecond id="w45"/>
        </elementSecondIds>
</triple>
</triples>
```

But also other relations can be expressed easily through additional triples:

```
<triple id="bus-accident.ont.dep.kaf#3" relation="part-of">
        <elementFirstIds label="event" comment="multiple agencies">
                <elementFirst id="w32"/>
                <elementFirst id="w33"/>
        </elementFirstIds>
        <elementSecondIds label="location" comment="Firefighters">
                <elementSecond id="w30"/>
        </elementSecondIds>
</triple>
<triple id="bus-accident.ont.dep.kaf#2" relation="part-of">
        <elementFirstIds label="event" comment="Mentone">
                <elementFirst id="w32"/>
        </elementFirstIds>
        <elementSecondIds label="location" comment="Highway 38">
                <elementSecond id="w36"/>
                <elementSecond id="w37"/>
        </elementSecondIds>
</triple>
```

Each triple has several attributes for additional information, such as the triple identifier, the heuristic or profile that generated the triple, the confidence value of the source of the triple, comments for each triple element, usually the words that form the expression and a label to indicate the type of expression.

The Triple module evaluates text mining output against a gold-standard in terms of precision, recall and f-measure. It compares the token ranges of the triple elements and the relation between them:

- Precision = nCorrect system triples/n gold standard triples
- Recall = nCorrect system triples/nr of system triples
- F-measure = 2 * (P*R)/(P + R)

The system will produce an excel file with the results. The next table shows some basic statistics for the gold-standard and system file, such as the number of triples, the average number of tokens per triple, whether or not the system triples are covering the same range of tokens as specified, how many unique triples there are (duplicates are removed).

Next, the table shows the degree to which the tokens of the first and second elements of the files match. Next it gives precision and recall just for the first and second elements. Finally, the table shows the precision, recall and f-measure for the triples, combining the token ranges of the elements and the relation between them. The partial identifiers label indicates that it is sufficient to have a single overlapping token te have match between two elements (first or second). Possibly, systems can cheat by defining very long ranges of tokens which always match the gold-standard element in case of a partial match. For that purpose, we measure the average number of tokens in the gold-standard and the system output. Furthermore, a range of tokens can be read to limit the range of text that is evaluated. If a file with the range of tokens is omitted, the scope if text is based on the sentences that have been used for the gold standard. The second table shows the numbers split per relation. It also indicates the number of triples per relation and the proportion.

When you run the program in debug mode, evaluations are carried out by comparing the triples in four ways:

- all tokens and the relation exactly match

- all tokens match and the relation is ignored

- at least one token matches and the relation matches

- at least one token matches and the relation is ignored

By ignoring the relation, we can estimate the maximum result in case of a perfect relation match. Partial matches of tokens are usually more appropriate than exact matches. Often human annotators that produce the gold-standard disagree about the exact range of words that should be annotated. In the above case, the location can either be the full phase \it{o Highway 38 near Bryant Street in Mentone} or any subset of these words.

Debug mode also produces a log file and more feedback on the recall (missed triples) and precision (wrong relations) errors, including a contingency table.

Precision and recall figures		
Gold standard	11767.tag.trp	
	Nr. of Triples	470
	Average nr. of first element ids	1
	Average nr. of second element ids	2
System file	11767.kaf.kybot.xml.0.trp	
	Nr. of Triples in total	195
	Nr. of Triples in range	195
	Nr. of Triples out of range	0
	Average nr. of first element ids	1
	Average nr. of second element ids	1
	Number of unique Triples in scope	162
	Number of first elements represented in gold standard Triples	263
	Number of first elements represented in system Triples	53
	Number of correct first elements represented in system Triples	53
	Recall of first elements	0.201520913
	Precision of first elements	1
	Number of second elements represented in gold standard Triples	362
	Number of second elements represented in system Triples	49
	Number of correct second elements represented in system Triples	49
	Recall of second elements	0.135359116
	Precision of second elements	1
Partial identifiers and same relation		
	Nr. correct	10
	Precision	0.061728395
	Recall	0.021276596

tokenRangeFile:	**../data/kyoto/11767.tag.trp.first-element-token-scope**						
tokenRange:	407						
Results per relation							
Relation	Total gold	Proportion Gold	Total system	Proportion system	PartialIdExactRelation	Recall	Precision
has-in-scope	0	0.00%	3	1.00%	0	0	0
has-change-situation	0	0.00%	3	1.00%	0	0	0
destination-of	36	7.00%	0	0.00%	0	0	0
use-of	5	1.00%	9	4.00%	0	0	0
has-path	0	0.00%	2	1.00%	0	0	0
generic-location	14	2.00%	15	7.00%	1	7	6
source-of	13	2.00%	1	0.00%	0	0	0
instrument	2	0.00%	5	2.00%	0	0	0
elementSecond	2	0.00%	0	0.00%	0	0	0
product-of	3	0.00%	2	1.00%	0	0	0
part-of	1	0.00%	3	1.00%	0	0	0
purpose-of	9	1.00%	4	2.00%	0	0	0
q-location	0	0.00%	3	1.00%	0	0	0
patient	165	35.00%	47	24.00%	5	3	10
path-of	1	0.00%	0	0.00%	0	0	0
result-of	14	2.00%	1	0.00%	0	0	0
participant	0	0.00%	5	2.00%	0	0	0
has-state	47	10.00%	8	4.00%	0	0	0
state-of	22	4.00%	0	0.00%	0	0	0
done-by	83	17.00%	30	15.00%	4	4	13
simple-cause-of	53	11.00%	23	11.00%	0	0	0
Total	470		164		10	2	6

2. Overall process to obtain evaluation data

The Triple Evaluation has a number of functions that deal with the conversion from various formats to the triple format and the evaluation functions themselves. Together they form a workflow for creating evaluation data for systems. Figure-1 below shows an overview where we assume two different processes: one manual annotation and one automatic annotation of the same text (obviously, triples can be produced and compared in any other way such as folded-cross validation). At start, a set of documents needs to be tokenized. As can be inferred from the above description, the tokenization is the only minimal preprocessing required to use the triple evaluation module. Any comarison of triples requires that the tokenized input is the same. Likewise, the two processes branch-of from the tokenized text. In this figure, the tokenized text is represented as KAF (KYOTO annotation format, Bosma et al 2009) and its successor NAF (NLP Annotation Format, Soroa et al. 2013).[2] Other formats can be used as well.

The left branch assume that some annotation tool is used to manually annotate the texts according to some predefined tag set. The annotations are stored in some TAG format, which can be any format as long as it can be translated to the token layer. We assume that the annotation process can be cyclic and can involve analysis of the agreement across annotators based on an inter-annotator-report. The annotation then needs to be converted to the above triple format involving the tokens from the text and the relations defined in the tag set. Various annotation tools are available, some also include conversions to the triple format.

The right branch shows some automated process (rule-based, machine-learning) that interprets texts and represents the result in some format: layers in KAF or NAF or as separate tuples. Here the same constraint applies as for the manual annotation: the representation needs to be convertible to relations between ranges of tokens. We provide conversion for various structures for various formats. New ones can easily be added.

Once we have two sets of triples for the same tokenized source texts, we can compare them and generate some evaluation report.

2 KAF and NAF are layered standoff representations developed in the KYOTO and the NewsReader project respecively. The format is designed for representing the results of automatic text analysis in different layers that point back to previous layers (ultimately the token layer of the text). In this way each layer can remain relatively simple and it is easy to add new layers without having to change the other layers.

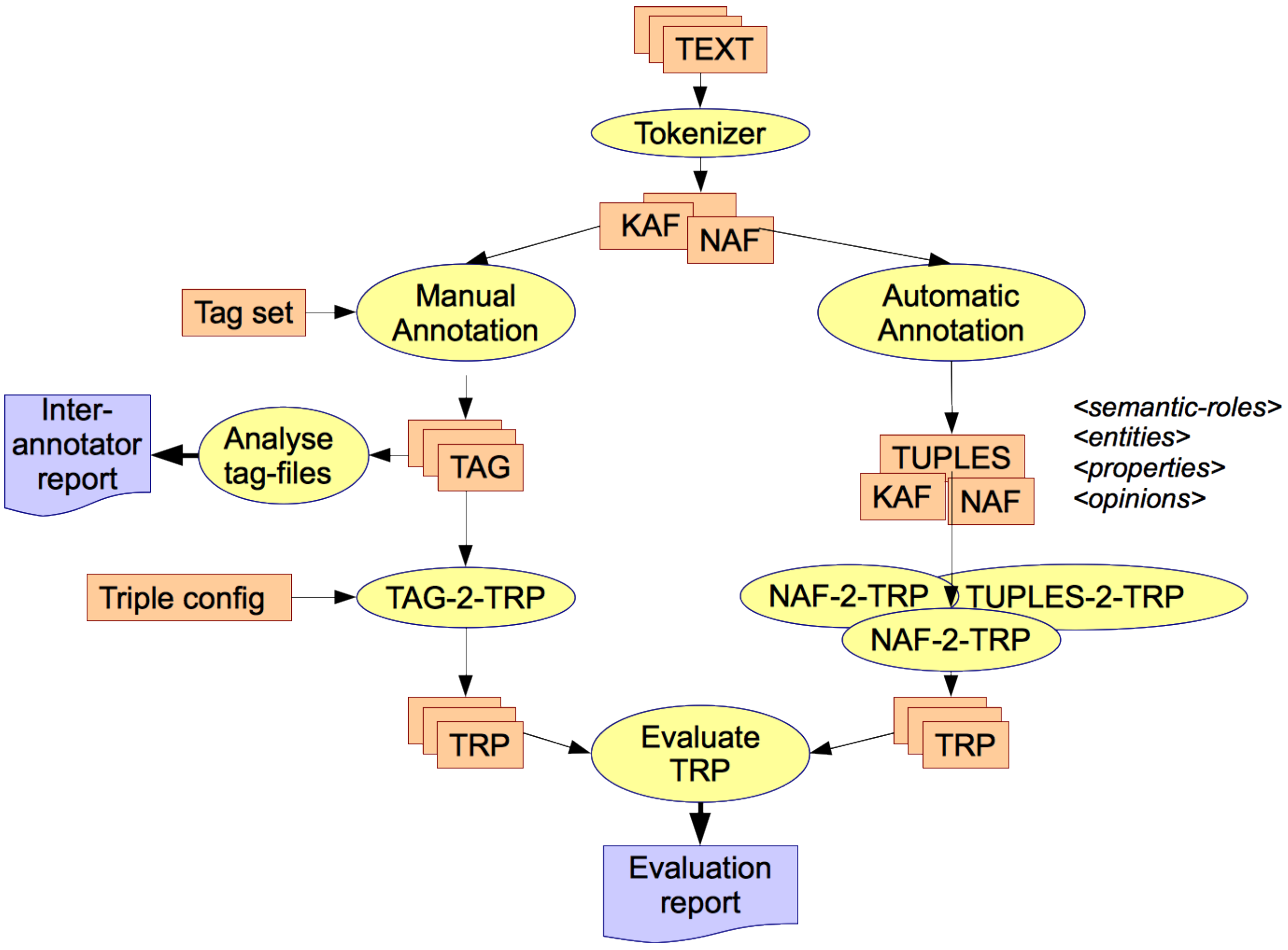

Figure 1: Evaluation work flow

3. Main function calls and program set-up

The program can be installed by unpacking the archive anywhere on the disk. The program is developed in Java 1.6. It requires Java1.5 runtime environment.

After unpacking the software you should get the following structure:

```
data
        kyoto
        opener
        nwr
java-doc
lib/TripleEvaluation-1.0-jar-with-dependencies.jar
scripts
        convert-relations.sh
        convertkybotoutputtotriplets.sh
        kaf-to-kyoto-baseline-triples.sh
        kaf-to-triples.sh
        kybotbaselineevaluation.sh
        kybotevaluation.sh
        openerevaluation.sh
        opinions.rel
        tuples-to-triples.sh
Readme.txt
Triple-evaluation-documentation.pdf (this document)
COPYING-GPL.TXT
LICENSESOFTWARE.TXT
```

The datat folder contains example data to test the program. Three cases are represented that represent data from three different projects:

KYOTO: www.kyoto-project.eu
OpeNER: http://www.opener-project.org/
NewsReader: http://www.newsreader-project.eu

The data in these folders is used in the example scripts.

The java-doc folder describes the Java API of the library that can be found in the lib folder. The scripts folder contains example of how to use the API on the data in the data folder.

In the next subsections, we explain the functions to convert data to the triple format and to run the evaluation.

3.1 Conversion functions to create triple files

Since different programs generate different formats for representing the annotations of text or the information that is mined, a separate conversion is needed from each respective output to the triple format.

3.1.1 Conversion of Kybot output to the triple format

The Kybot program is a module developed in the KYOTO project for extracting events, participants, their roles and time and place from text in 7 different languages. The output of the Kybot program is XML. An example is shown in the file data/kyoto/11767.kaf.kybot.xml. A fragment is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<kybotOut>
  <doc shortname="11767.mw.wsd.ne.kaf.reduced-to-migration.kaf.onto">
    <event eid="e92" target="t9018" lemma="fish" pos="V" synset="eng-30-01140794-v" rank="0.534691"
profile_id="generic_kybot_Vaccomplishment-Naccomplishment"/>
    <role rid="r170" event="e92" target="t9020" lemma="restoration" pos="N" rtype="patient" synset="eng-30-00268557-
n" rank="0.175324" profile_id="generic_kybot_Vaccomplishment-Naccomplishment"/>
    <role rid="r174" event="e92" target="t9019" lemma="passage" pos="N" rtype="patient" synset="eng-30-00201058-n"
rank="0.101492" profile_id="generic_kybot_Vaccomplishment-Naccomplishment"/>
    <role rid="r179" event="e92" target="t9019" lemma="passage" pos="N" rtype="patient" synset="eng-30-03895293-n"
rank="0.118576" profile_id="generic_kybot_Vaccomplishment-Nphysical-object-OR-matter"/>
```

This XML structure has 1 event (92) and 3 role elements that are connected to the event through the event identifier "92". For each elements various attributes are given, making the format very specific and idiosyncratic. The target attribute for example points to a term layer in KAF which refers further to the tokens of the text.

The KybotOutputToTriples class can be used to convert this structure to triples, where it takes the events as the first element, the roles as the second element and the rtype attribute as the relation. The term identifiers are converted to the token indentifier. It needs the original KAF file for the converting term identifiers to token identifiers.

Main class:

- vu.tripleevaluation.conversion.KybotOutputToTriples

Arguments:

- arg1: the output of the Kybots that extract events in the KYOTO system
- arg2: the KAF file from which the Kybot output is generated
- arg3: the threshold for the WSD score of events and roles, if set to 0 all output is taken, if set to 100 only the highest scoring interpretation in case of competition. All other values are proportional to the highest score.

The function takes 3 obligatory arguments. The program is demonstrated by the script: scripts/convertkybotoutputtotriplets.sh. The result is stored in data/11767.kaf.kybot.xml.0.trp and data/11767.kaf.kybot.xml.60.trp. The result looks as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<triples>
<triple id="e92" profile_id="generic_kybot_Vaccomplishment-
Naccomplishment,generic_kybot_Vaccomplishment-N-Naccomplishment" relation="patient">
        <elementFirstIds comment="fish">
                <elementFirst id="w10948"/>
        </elementFirstIds>
        <elementSecondIds comment="restoration">
                <elementSecond id="w10950"/>
        </elementSecondIds>
</triple>
<triple id="e92" profile_id="generic_kybot_Vaccomplishment-Naccomplishment" relation="patient">
        <elementFirstIds comment="fish">
                <elementFirst id="w10948"/>
        </elementFirstIds>
        <elementSecondIds comment="passage">
                <elementSecond id="w10949"/>
        </elementSecondIds>
</triple>
<triple id="e92" profile_id="generic_kybot_Vaccomplishment-Nphysical-object-OR-matter"
relation="patient">
        <elementFirstIds comment="fish">
                <elementFirst id="w10948"/>
        </elementFirstIds>
        <elementSecondIds comment="passage">
                <elementSecond id="w10949"/>
        </elementSecondIds>
</triple>
```

3.1.2 Conversion of KAF Kybot tuples to triples

Another KYOTO module generates tuples of any structure from KAF files rather than just the events and roles. Tuples consist of any number of elements with any name. To convert them to tuples, one of the element needs to be the parent and all the other elements will become children. When converted to a triples, a separate triple is generated for each pair of parent and child element. The tuple identifier is used as the triple identifier to trace back triples to the tuple from which they are derived. Below is a fragment of a tuple file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kafkybot-results>
  <tuples source="bus-accident.ont.dep.kaf">
    <tuple id="1" profile="agent-1-sem" profileConfidence="30" sentenceId="s2">
      <!--Firefighters from multiple agencies responded to Highway 38 near Bryant Street in Mentone about 6:30 p.m. .-->
      <event concept="eng-30-00717358-v" confidence="0.703748" lemma="respond" mention="t33" pos="VBD"
tokens="w34"/>
      <participant concept="eng-30-10091651-n" confidence="1.0" lemma="firefighter" mention="t29" pos="NNS"
reference="ExtendedDnS.owl#social-object" role="agent" tokens="w30"/>
    </tuple>
    <tuple id="3" profile="agent-1-sem" profileConfidence="30" sentenceId="s1">
      <!--Several people died and 27 people were injured on Sunday when a private charter tour bus coming down a mountain
road collided with an SUV and another car .-->
      <event concept="eng-30-00358431-v" confidence="0.662059" lemma="die" mention="t3" pos="VBD" tokens="w3"/>
      <participant concept="eng-30-08160276-n" confidence="0.0567295" lemma="people" mention="t2" pos="NNS"
reference="ExtendedDnS.owl#social-object" role="agent" tokens="w2"/>
    </tuple>
```

The tuples have various attributes as well but include pointers to the terms using the mention attribute and pointers to the tokens through the tokens attribute.

Main class:

- vu.tripleevaluation.conversion.ConvertTuplesToTriples

Arguments:

--tuple-file	<path to the file or folder with the tuples>
--first-element	<the name of the element in the tuple that will become the first element in the triples, all other elements will become a child>
--extension	<in case a folder with tuple files is given , then the file extension of the tuple files can be specified to filter the files to be >

This program is shown by the script: scripts/tuples-to-triples.sh. The above text is converted to the following triples:

```
?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<triples>
<triple id="bus-accident.ont.dep.kaf#1" profile_id="agent-1-sem" profile_confidence="30" relation="agent">
        <elementFirstIds label="event" comment="respond">
                <elementFirst id="w34"/>
        </elementFirstIds>
        <elementSecondIds label="participant" comment="firefighter">
                <elementSecond id="w30"/>
        </elementSecondIds>
</triple>
<triple id="bus-accident.ont.dep.kaf#3" profile_id="agent-1-sem" profile_confidence="30" relation="agent">
        <elementFirstIds label="event" comment="die">
                <elementFirst id="w3"/>
        </elementFirstIds>
        <elementSecondIds label="participant" comment="people">
                <elementSecond id="w2"/>
        </elementSecondIds>
</triple>
```

3.1.3 Conversion of KAF to triples

Triples can be extracted directly from certain layers in the KAF structure. These functions are hard-coded.

Main class:

- vu.tripleevaluation.conversion.ConvertKafToTriples

Arguments:

--kaf-file	<path to a single kaf file>
--kaf-folder	<path to a folder with kaf files>
--extension	[OPTIONAL] <file extension of the KAF files to be considered, to be used with the --kaf-folder option>
--intersect	[OPTIONAL] <only use opinions that have targets or holders that intersect with properties or entities>
--opinion	[OPTIONAL] <opinion layer is converted to triples>
--entity	[OPTIONAL] <entity layer is converted to triples>
--property	[OPTIONAL] <property layer is converted to triples>
--term-sentiment	[OPTIONAL] <sentiments at the term layer are converted to triples>
--srl	[OPTIONAL] <semantic role layer is converted to triples>

This program is shown by the script: scripts/kaf-to-triples.sh

3.1.4 Baseline triples for KYOTO

The package includes a function to extract a baseline from a KAF file. The baseline creates triples between all the heads of constituents (chunks), taking one as the event and all the others as the roles. It assumes that a chunk layer is present in the KAF representation.

Main class:

vu.tripleevaluation.kyoto.BaseLineForChunks

The first argument is the KAF file, the second argument is optional and can be used to name a default relation, e.g. patient. The use of this function is shown in scripts/kybotbaselineevaluation.sh.

3.2. Translation of triple relations

In some cases, relations in triples need to be adapted, e.g. because they are too fine-grained.

Main class:

vu.tripleevaluation.conversion.TripleRelationConversion

Arguments:

--triples	<path to the triple file>
--relation-mapping	<path to a text file on each line the source relation+tab+targe relation for translation>

3.3. Evaluation of triple files

3.3.1 Evaluating two triple files

Main class:

- vu.tripleevaluation.evaluation.EvaluateTriples

Arguments:

--gold-standard-triples	file with gold standard triples
--system-triples	file with system triples
--token-range (optional)	file with tokens for the events to be covered
--ignore-element-second	(optional) lumps differentiated second elements into one single typed
--ignore-relations	(optional) relation labels are ignored for matching
--skip-time-and-location	(optional) TIME and LOCATION relations are ignored

Output:

- xls file with statistics and recall & precision for the system file.

The evaluation is shown through script/kybotevaluation.sh. This script takes 11767.tag.trp as the gold-standard and the above created system files as the system triples. The result is an xls file (see above). This file has details about the statistics of the triples generated and precision/recall and f-measure. The results are given per relation and overall per file. Also some other statistics are given, such as total number of triples, nr of first elemenets, second elements, average number of first and second element identifiers in the triples (if this deviates too much from the gold standard the evaluation is not valid).

The tokenrange indicates which tokens are considered. This is used when only a small part of the document is annotated while systems typically extract relations from the complete text. By giving a range of tokens the evaluation can be restricted to triples for that range only. You can specify any range of tokens. If the first element tokens of the system are in that range, it is considered. Typically, token-ranges are provided for gold-standards for the first element or for the sentences that include these.

If the class vu.tripleevaluation.evaluation.EvaluateTriplesDebug is used instead, it generates details on the type of analysis, such as the precision for each profile, per relation and different types of matches: exact and partial identifiers, exact relation and ignoring the relations. It also generates a log file listing all missed triples (to improve recall), all correct matches and a confusion matrix for the profiles. There is also a log file. This log file contains a list of all the triples that were missed by the system, and errors generated in non-missed relations, e.g.:

```
* Not covered Triples:458

Sorted Triples:458
```

... followed by the sorted list of triples that were not detected

```
* Frequency of missed Triple relations:
```

.... followed by a table with frequency of missed triples per relation

```
destination-of      36
use-of              5
generic-location    13
source-of           13
instrument          2
elementSecond       1
product-of          3
part-of             1
purpose-of          9
patient             160
path-of             1
result-of           12
has-state           47
state-of            22
done-by             80
simple-cause-of     53
```

```
* Missed Triples as table relations:
```

.... followed by the same missed triples in simple format per line

```
commercial and recreational fisheries:use-of:The Chesapeake Bay and its tributaries
Drive less:use-of:your car
Use:use-of:phosphorus-free dish detergent
most beneficial use:use-of:Forests
passage:use-of:fish
lowered:state-of:by 11 percent
desired health:state-of:38 percent
```

* Triples with partial ID match and correct relations: etc.

* Triples with partial ID match but wrong relation: etc.

* Frequency of wrong Triple relations:

patient	5
simple-cause-of	3
purpose-of	2

To run the program in debug mode use:

- vu.tripleevaluation.evaluation.EvaluateTriplesDebug

3.3.2. Evaluating two folders with triple files

This program will compare a folder with system triple files with a folder with gold-standard triple files.

Main class

vu.tripleevaluation.evaluation.EvaluateTripleFolders

Arguments:

--gold-standard-triples	<path to the folder with the gold-standard triples
--system-triples	<path to the folder with the system triples>
--key	[OPTIONAL]<any string to name the evaluation result file>
--ignore-file-suffix	[OPTIONAL]<number of characters that are ignored to compare a gold-standard file with a system file. If left out only identical file names are compared, otherwise they need to match expect for specified substring length>
--ignore-element-second	[OPTIONAL]<only the first element of the triples is considered>
--skip-time-and-location	[OPTIONAL]<Time and location included in the KYOTO based triples are ignored>
--relation-filter	<path to a text file listing the relations that need to be considered. This can be used to limit the evaluation to certain relations only. Each relation is listed on a separate line

The function will create a subfolder in the system triple folder with a date stamp and place and overview XLS file in that folder. This function is shown in the script /scripts/openerevaluation.sh. The script shows how system generated opinions in hotel reviews are compared with manually annotated reviews. They share the beginning of the file names but differ in the suffixing (name ending). The option --ignore-file-suffix is used to indicate what part of the file name should not be matched for files to be compared.

References

Bosma, Wauter, Piek Vossen, Aitor Soroa, German Rigau, Maurizio Tesconi, Andrea Marchetti, Monica Monachini and Carlo Aliprandi: “KAF: a generic semantic annotation format”, in Proceedings of the 5th International Conference on Generative Approaches to the Lexicon GL 2009, Pisa, Italy, September 17-19, 2009

Soroa, Aitor, Antske Fokkens, …...2013