Evaluation Module

Deliverable D3.2

Version FINAL

Authors: Piek Vossen[1], Agata Cybulska[1], Sara Tonelli[2], Rachele Sprugnoli[2]
Affiliation: (1) VUA, (2) FBK

Building structured event indexes of large volumes of financial and economic data for decision making
ICT 316404

Grant Agreement No.	316404
Project Acronym	NEWSREADER
Project Full Title	Building structured event indexes of large volumes of financial and economic data for decision making.
Funding Scheme	FP7-ICT-2011-8
Project Website	http://www.newsreader-project.eu/
Project Coordinator	Prof. dr. Piek T.J.M. Vossen VU University Amsterdam Tel. + 31 (0) 20 5986466 Fax. + 31 (0) 20 5986500 Email: piek.vossen@vu.nl
Document Number	Deliverable D3.2
Status & Version	FINAL
Contractual Date of Delivery	July 2013
Actual Date of Delivery	July 23, 2013
Type	Prototype
Security (distribution level)	Public
Number of Pages	34
WP Contributing to the Deliverable	WP03
WP Responsible	FBK
EC Project Officer	Sophie Reig
Authors: Piek Vossen[1], Agata Cybulska[1], Sara Tonelli[2], Rachele Sprugnoli[2]	
Keywords: Evaluation metrics, evaluation system, triple-based evaluation, coreference	
Abstract: Here comes the Abstract.	

Table of Revisions

Version	Date	Description and reason	By	Affected sections
0.1	June 2013	Document created. Draft of deliverable structure	Sara Tonelli	
0.2	June 2013	Description of triple-based evaluation	Piek Vossen	
0.3	June 2013	Description of coreference evaluation	Agata Cybulska	
x.0	date	approval by project manager	who	-

Executive Summary

executive summary text....

Contents

List of Tables

1 Introduction

2 Evaluating generic triples

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 *responded*, involves a participant, *firefighters from multiple agencies*, a location, *Highway 38 near Bryant Street in Mentone*, and a time expression, *6:30 p.m.*. These elements also have internal structures, such as 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 following 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>
```

In this representation, the `participant`, `location` and `time` elements are grounded to tokens of the text and indirectly connected to the `event` element that governs the other elements.

Any structure can be decomposed in a set of basic relations between expressions. Given that within NewsReader several information layers will be added to the input news, an evaluation approach which is flexible and extensible is needed. Therefore, we have implemented a **Triple Module** in order 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 represents the parent element, governing a second element that is a child. In the case of the above

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

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: numCorrect system triples / numSystem triples
- Recall: numCorrect system triples / numGoldStandard triples
- F-Measure: $2 * (P * R)/P + R$

The system will produce an excel file with the results. We show in Table 1 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 to have a 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 of text is based on the sentences that have been used for the gold standard.

Table 2 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

Precision and Recall Figures		
Gold standard	11767.tag.trp	
	N. 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	
	N. of triples in total	195
	N. of triples in range	195
	N. 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

Table 1: Evaluation statistics: An example

tokenRangeFile: 11767.tag.trp.first-element-token-scope
tokenRange: 407

Results per relation
Results per relation

Relation	Total gold	% Gold	Total system	% system	PartialIdtRelation	Exact-	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

Table 2: Evaluation statistics for each relation

- 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 *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.

3 Evaluating coreference

Coreference is the relation that links textual expressions, i.e. mentions, referring to the same entity or event, creating together chains of coreferring mentions. Such mentions may appear in the same text (*intra-document coreference*) or in different resources (*cross-document coreference*).

Over the years different scoring measures have been developed but, at present, there is no agreement on a standard measure for coreference resolution evaluation. The four most used metrics are: MUC ([Vilain *et al.*, 1995]), B^3 ([Bagga and Baldwin, 1998]), ACE-Value ([Doddington *et al.*, 2004]), and CEAF ([Luo, 2005]). More recently, a new measure named BLANC ([Recasens and Hovy, 2011]) has been proposed with the aim of overcoming the weak points of the existing measures. The main difference among the above mentioned metrics relies on the coreference resolution model they are based on: link-based metrics, such as MUC and BLANC, work in terms of pairs of coreferent mentions whereas class-based metrics, such as B^3, CEAF, and ACE-Value, treat entities as clusters. Finally, there is the CoNLL F1 metric [**?**] which is calculated as an average of MUC, B-CUBED and CEAF.

The various scoring algorithms are presented below:

- MUC was defined within the Message Understanding Conference evaluation tasks [Hirschman, 1997] on coreference resolution. It is computed by counting the number of common links, i.e. pairs of coreferent mentions, between the gold standard and the system output. Precision corresponds to the number of common links divided by the number of links in the system output and recall is the number of common links divided by the number of links in the gold standard. The major drawbacks are that MUC metric ignores entities which are mentioned only once (singletons), given that no link can be found for these entities. The metric favors systems producing fewer entities, while it penalizes less the systems that wrongly merge entities.

- B^3 computes precision and recall for all individual mentions (thus it takes into account entities which are mentioned only once) and then combines the weighted sum of these individual precisions and recalls to produce the final score. Variants of this metric have been proposed in [Stoyanov *et al.*, 2009], [Rahman and Ng, 2009] and [Cai and Strube, 2010] in order to solve the shortcomings of B^3 such as the fact that, when many single-mention entities are present, the score increases approaching 100%.

- ACE-Value is used to measure the overall performance of systems participating in the Automatic Content Extraction program evaluation campaigns. This metric is computed by counting the number of errors produced by the system in the recognition of mentions and entities and normalizing this sum against the cost of a system that does not output any entity. The result of the normalization is then subtracted from 1 and a system could get even a negative score. This metric is very task-specific (restricted to scoring the set of ACE entities) and hard to interpret, given that each type of error has an associated cost that has changed across successive evaluations.

- CEAF (Constrained Entity Aligned F-Measure) is based on the one-to-one alignment of system entities to gold standard entities; this means that a gold standard entity can be aligned with at most one system entity, and vice-versa. The alignment is calculated applying a similarity metric at the level of mentions (mention-based CEAF or CEAFM) or at the level of entities (entity based CEAF or CEAFE) for each set of mentions: the best alignment is used to compute recall, precision and F-measure. The two major known weaknesses of this metric are *i)* the fact that a correct coreference link might be ignored during the alignment phase if the entity involved in the link doesn't have a counterpart in the gold standard and *ii)* that all the entities are weighted equally despite the number of mentions they contain. Variants of CEAF have been proposed in [Rahman and Ng, 2009] and [Cai and Strube, 2010].

- BLANC (BiLateral Assessment of Noun-phrase Coreference) adapts the Rand index [Rand, 1971], originally developed for the evaluation of clustering methods, to coreference. It computes recall, precision and F-measure both for coreference links (i.e. links between every two mentions that corefer) and for non-coreference links (i.e. links between every two mentions that do not corefer) and averages them to obtain the final score.

- CoNLL-F1 metric was used for the evaluation of the shared task on coreference resolution within the CoNLL-2011 [**?**] . This evaluation measure is based on the MELA score (for Mention, Entity, and Link Average, [**?**] which is a combination of a number of evaluation metrics. MELA is calculated as a weighted average of F scores of three measures, each focusing on a different dimension of coreference evaluation: link based MUC, mention based B-CUBED and entity based CEAF. For scoring of systems within the CoNLL-2011 shared task an unweighted mean of MUC, B-CUBED and CEAFE was employed.

Within the "Coreference Resolution in Multiple Languages" task at SemEval-2010 four different metrics (MUC, B-CUBED, CEAF and BLANC) have been used during the evaluation. This comparison showed that the rankings of systems considerably vary depending on the chosen metric highlighting the need of defining a new way to evaluate coreference.

Although the Triple module presented in Section 2 is flexible enough to cover several types of relations, it is not optimal for coreference, because it requires more sophisticated metrics than just triple-based ones. In fact, coreference should be rather represented as a chain than as a binary relation, and transitivity should be taken into account (e.g. if A corefers with B and B corefers with C, a system that marks A as coreferring with C may be correct). Therefore, we decided to have two separate evaluation modules within NewsReader: one which is a generic one for any kind of binary relations (Section 4), and one that is specific to coreference (Section **??**).

4 Module for Triple-based Evaluation

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

[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.

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.

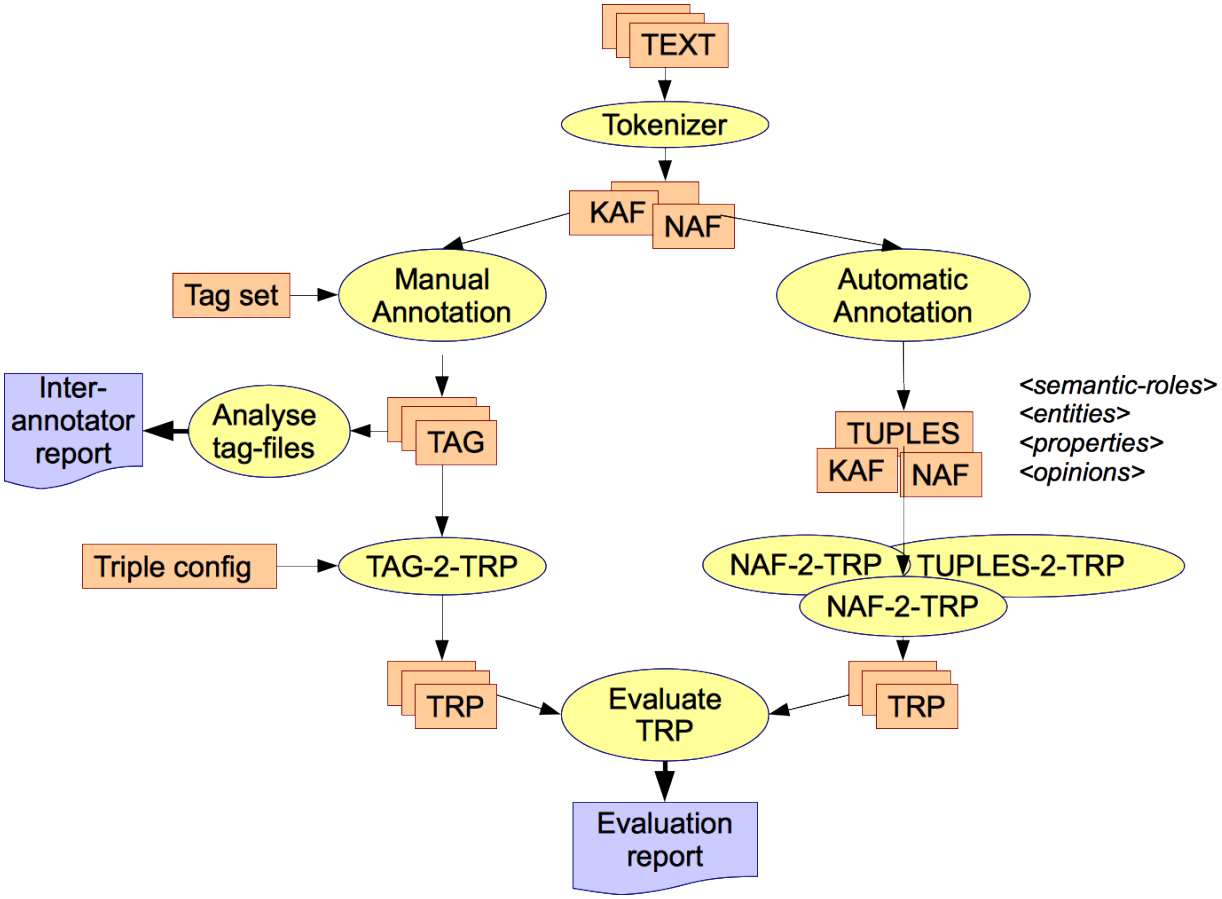

Figure 1: Evaluation workflow

4.1 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
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.

4.2 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.

4.2.1 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>
```

4.2.2 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>
```

4.2.3 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

4.2.4 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.

4.3 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
```

4.4 Evaluation of triple files

4.4.1 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, number of first elements, 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.

Table 3 shows an example of the output tables generated per relation, comparing a system file with a gold-standard file.

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
```

Relation	Total gold	Prop. Gold	Total system	Prop. system	Matches	Recall	Precision
has-in-scope	0	0%	3	1%	0	0	0
has-change-situation	0	0%	3	1%	0	0	0
destination-of	36	8%	0	0%	0	0	0
use-of	5	1%	9	4%	0	0	0
has-path	0	0%	2	1%	0	0	0
generic-location	14	3%	15	7%	1	7	6
source-of	13	3%	1	0%	0	0	0
instrument	2	0%	5	2%	0	0	0
product-of	3	1%	2	1%	0	0	0
part-of	1	0%	3	1%	0	0	0
purpose-of	9	2%	4	2%	0	0	0
q-location	0	0%	3	1%	0	0	0
patient	165	35%	47	24%	5	3	10
path-of	1	0%	0	0%	0	0	0
result-of	14	3%	1	0%	0	0	0
participant	0	0%	5	2%	0	0	0
has-state	47	10%	8	4%	0	0	0
state-of	22	5%	0	0%	0	0	0
done-by	83	18%	30	15%	4	4	13
simple-cause-of	53	11%	23	11%	0	0	0
Total	468		164		10	2	6

Table 3: Output of the triple evaluation per relation

```
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

4.4.2 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.

4.4.3 Evaluating unary annotations

Even though the triple module is designed for complex relational data, it is possible to use it for unary annotations of tokens in text, such as part-of-speech tags, entity tags or any other type of marks. In that case, the elementSecondIds can be left empty. Below is an example of a polarity annotation of words, where the range of tokens is given for the elementFirstIds and the value for the relation attribute expresses the annotation:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<triples>
<triple id="t_6" relation="intensifier" ">
       <elementFirstIds comment="very">
              <elementFirst id="w_6"/>
       </elementFirstIds>
       <elementSecondIds/>
</triple>
<triple id="t_7" relation="positive">
       <elementFirstIds comment="excited">
              <elementFirst id="w_7"/>
       </elementFirstIds>
       <elementSecondIds/>
</triple>
<triple id="t_17" relation="negative">
       <elementFirstIds comment="very bad">
              <elementFirst id="w_16"/>
              <elementFirst id="w_17"/>
       </elementFirstIds>
       <elementSecondIds/>
</triple>
</triples>
```

This example shows that single tokens and token ranges can be annotated. The evaluation of such structure is exactly the same as for triples with both first and second scope defined. In this case the elementSecondIds always match, given a match of the elementFirstIds.

Table 4 shows the results for triples that represent sentiment values for expressions without the elementSecondIds specified. The table has the same structure as the previous relational triples.

Relation	Total gold	Prop. Gold	Total system	Prop. system	Matches	Recall	Precision
positive#2	81	13%	79	6%	17	20	21
positive#	248	40%	289	21%	13	5	4
negative	0	0%	228	16%	0	0	0
intensifier	0	0%	131	9%	0	0	0
positive#3	0	0%	7	0%	0	0	0
negative#-2	0	0%	49	3%	0	0	0
negative#-1	284	46%	224	16%	20	7	8
negative#-3	0	0%	3	0%	0	0	0
positive#4	0	0%	1	0%	0	0	0
shifter	0	0%	49	3%	0	0	0
positive	0	0%	346	25%	0	0	0
Total	613		1406		50	8.16	3.56

Table 4: Output of the triple evaluation per relation

5 NewsReader Module for coreference evaluation

As explained in section 4, there is no agreement at present on a standard measure for coreference resolution evaluation. We therefore developed a module that applies all four models to any coreference data set. The module is designed to handle cross-document co-reference sets. Cross-document co-references complicate the process because the identifiers of the sets and their mentions need to be unique across different documents and sources. We defined a coreference representation format within the Grounded Annotation Framework (GAF)[3] to uniquely identify coreference sets and their mentions

The format for representing coreferences is adapted to the distinction between mentions and instances as defined in GAF. This means that all references to mentions in the text use URIs that can be resolved in the representation of the text. Every co-reference set receives a unique instance id that is also resolvable. We applied these principles to the EECB corpus for cross-document coreferences. EECB is an extension by [**?**; **?**] of the ECB corpus [**?**]. Below is an example of part of the EECB co-reference data that is made GAF compliant by creating reference to identifiers in KAF files:

```
<?xml version="1.0" encoding="UTF-8"?>
<coreferences version="v1" xmlns:eecb1.0="http://faculty.washington.edu/
    bejan/data/ECB1.0.tar.gz">
   <coref cid="eecb1.0:1_1">
      <target id="eecb1.0:1/1.eecb.kaf_stanford-parser-en#t22" head="yes"/>
   </coref>
   <coref cid="eecb1.0:1_2">
      <target id="eecb1.0:1/1.eecb.kaf_stanford-parser-en#t28" head="yes"/>
   </coref>
   <coref cid="eecb1.0:1_3">
      <target id="eecb1.0:1/1.eecb.kaf_stanford-parser-en#t37" head="yes"/>
      <target id="eecb1.0:1/2.eecb.kaf_stanford-parser-en#t6" head="yes"/>
      <target id="eecb1.0:1/3.eecb.kaf_stanford-parser-en#t25" head="yes"/>
      <target id="eecb1.0:1/7.eecb.kaf_stanford-parser-en#t7" head="yes"/>
      <target id="eecb1.0:1/7.eecb.kaf_stanford-parser-en#t16" head="yes"/>
      <target id="eecb1.0:1/7.eecb.kaf_stanford-parser-en#t26" head="yes"/>
   </coref>
   <coref cid="eecb1.0:1_4">
      <target id="eecb1.0:1/1.eecb.kaf_stanford-parser-en#t67" head="yes"/>
      <target id="eecb1.0:1/2.eecb.kaf_stanford-parser-en#t3" head="yes"/>
   </coref>
</coreferences>
```

In this example, we first provide a namespace definition to the ECB source file. References to coreference sets (*coref*) and to mentions within different files (*target*) use this namespace. The ECB corpus consists of topics represented through numbered subfolders. Within each subfolder, e.g. folder *1*, several files are given with news articles on the same

[3] groundedannotationframework.org

topic. Cross-document co-reference relations hold within a topic. To make the co-reference sets unique, we created an identifier that combines the topic and the entity or event identifier. The identifier *1_3* refers to the third event in the first topic. Within the coref set, there are 6 mentions shown in 4 different files. Each mention is identified by the target id, using the namespace, the folder name and the filename in the folder. The file name is extended with the processor that created the element, in this case the English Stanford parser, and after the "#", the actual local identifier is given.

The current version of the coreference evaluation module is implemented in python, and can be divided into four clusters of functions:

1. functions converting the coreference sets from the above xml format into a python list

2. functions generating equivalence classes based on element overlap amongst (sub-)chains

3. a function generating relative partitions of key and response, based on their equivalence classes

4. five main functions for coreference evaluation.

We specify the function calls and output for each function in more detail below.

(1) functions converting the coreference sets from the above xml format into a python list:

```
class My_Corefs
function call: My_corefs(xml file with system output)

Output format: a python list with sub-lists containing word ids of co-refering items
```

(2) functions generating equivalence classes based on element overlap amongst (sub-)chains

```
function call:
generate_equivalence_classes(python list object with sublists
                        of co-refering word id or with singleton sublists)

Python list object as input for generate_equivalence_classes:
[ ["WORD ID1"],  ["WORD ID1", "WORD ID2"], ["WORD ID2", " WORD ID3"] ]

Output format: a python list with sub-lists containing word ids of co-refering items
```

(3) a function generating relative partitions of key and response, based on their equivalence classes

```
Function call:
generate_rel_partition(equivalence classes of the key,
                                    equivalence classes of the response)

Output: relative partition of response given key,
             relative partition of key given response
```

(4) five main functions for coreference evaluation.

This is the core of the module. Each function corresponds to a coreference evaluation metric. These functions take as input the equivalence classes of the key and of the response as well as in some cases also their relative partitions (depending on the measurement) and they produce as output a file in *.csv format, specifying recall, precision and F-score for each evaluation metric.

- calculate_MUC(equivalence classes, relative partitions, different combinations for recall and precision)

  ```
  Function calls:
  recall MUC = calculate_MUC(equivalence classes key,
        relative partition of key given response)
  precision MUC = calculate_MUC(equivalence classes  response,
             relative partition of response given key)
  ```

- calculate_B3(equivalence classes key, equivalence classes response, relative partition of response given key, relative partition of key given response)

  ```
  Function call:
  precision B3, recall B3 = calculate_B3(equivalence classes key,
      equivalence classes  response,
      relative partition of response given key,
      relative partition of key given response)
  ```

- calculate_CEAF(equivalence classes key, equivalence classes response)

  ```
  Function call:
  precision CEAFm, recall CEAFm, fscore CEAFm,
  precision CEAFe,  recall CEAFe, fscore CEAFe =
  calculate_CEAF(equivalence classes key,equivalence classes  response)
  ```

- calculate_BLANC(equivalence classes key, equivalence classes response)

  ```
  Function call:
  precision BLANC, recall BLANC, fscore BLANC = calculate_BLANC
  (equivalence classes key, equivalence classes  response)
  ```

- calculate_MELA(fscore-MUC,fscore-B3,fscore-CEAFe)

```
Function call:
fscore MELA = calculate_MELA(fscore-MUC,fscore-B3,fscore-CEAF)}
```

Table 5 shows the output of the evaluation as it is generated in a csv file.

Metrics	Recall	Precision	F-measure
MUC	0.583333333	0.368421053	0.451612903
B3	0.811403509	0.615789474	0.700190858
CEAFm	0.526315789	0.526315789	0.526315789
CEAFe	0.43453555	0.594627595	0.502129969
BLANC	0.666814159	0.579335017	0.602375566
MELA			0.551311243

Table 5: Output of the evaluation metrics

Currently, there is a discussion in the research community to provide a single measure and a single module for evaluating co-references. The work on this has just started. When the results are available, we will consider the new metrics and module for our purposes.

6 Conclusions

In this deliverable, we describe the generic evaluation metrics and modules that we will use in NewsReader. The modules are very generic and can be used to evaluate any annotation of text. In the case of the triple evaluation, more complex structures need to be converted to triples. Some functions have been provided. In the near future, we will add function to convert other formats that are used in NLP, such as CoNLL, to the triple format. The same holds for the co-reference module that we developed.

References

[Bagga and Baldwin, 1998] Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, 1998.

[Cai and Strube, 2010] Jie Cai and Michael Strube. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 28–36, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Doddington *et al.*, 2004] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The automatic content extraction (ace) program - tasks, data, and evaluation. pages 837–840, 2004.

[Hirschman, 1997] Lynette Hirschman. MUC-7 coreference task definition. In *Proceedings of the 7th Message Understanding Conference*, 1997. `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/co_task.html`.

[Luo, 2005] Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[Rahman and Ng, 2009] Altaf Rahman and Vincent Ng. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977, 2009.

[Rand, 1971] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[Recasens and Hovy, 2011] Marta Recasens and Eduard H. Hovy. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510, 2011.

[Stoyanov *et al.*, 2009] Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. Conundrums in noun phrase coreference resolution: making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 656–664, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Vilain *et al.*, 1995] Marc B. Vilain, John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *MUC*, pages 45–52, 1995.