ANNALIST: ANNotation ALIgnment and Scoring Tool

George Demetriou, Robert Gaizauskas, Haotian Sun and Angus Roberts

Department of Computer Science, University of Sheffield

211 Portobello Street, Sheffield S1 4DP, UK

{G.Demetriou, R.Gaizauskas, H.Sun, A.Roberts}@dcs.shef.ac.uk

Abstract

In this paper we describe ANNALIST (Annotation, Alignment and Scoring Tool), a scoring system for the evaluation of the output of semantic annotation systems. ANNALIST has been designed as a system that is easily extensible and configurable for different domains, data formats, and evaluation tasks. The system architecture enables data input via the use of plugins and the users can access the system's internal alignment and scoring mechanisms without the need to convert their data to a specified format. Although developed for evaluation tasks that involve the scoring of entity mentions and relations primarily, ANNALIST's generic object representation and the availability of a range of criteria for the comparison of annotations enable the system to be tailored to a variety of scoring jobs. The paper reports on results from using ANNALIST in real-world situations in comparison to other scorers which are more established in the literature. ANNALIST has been used extensively for evaluation tasks within the VIKEF (EU FP6) and CLEF (UK MRC) projects.

1. Introduction

There has been an explosion of interest in text mining systems in intensively researched fields such as bioinformatics and the Semantic Web. We use the term *semantic annotation system* to denote any information extraction or text mining system that is used to produce information, either inline or in a separate file, in the form of *annotations*, i.e. metadata, that are used to describe or characterise entities, relations and events in natural language texts.

The increased interest in semantic annotations systems brings with it an increased requirement *to evaluate* such systems, both for purposes of system development and for assessing the utility of systems for specific applications or for comparing competing systems. Existing tools for the evaluation of text mining systems are usually available either through competitions such as MUC (MUC, 1995) or ACE (ACE, 2007) or as part of bigger Natural Language Processing (NLP) architectures such as GATE (Cunningham et al, 2002) or Alembic (Day et al, 1997). With the exception of MUC and ACE scorers developed at DARPA and NIST respectively, there is generally a lack of available tools for the evaluation of semantic annotations of text. However, the MUC and ACE scorers, although extremely useful, still have limitations. The MUC scorer, developed over 10 years ago, is tailored to the MUC information extraction tasks (named entity recognition, relation and scenario extraction, coreference resolution). It admittedly performs quite well for these tasks, but requires input in SGML, and is quite difficult to extend. For tasks other than MUC, users have to convert their data to MUC scorer's format which may not be always desirable. The ACE scorer is a more recent tool (latest version dated 2006) but it also has problems of extensibility and maintainability. The use of cost weighting for scoring annotations in ACE may be desirable for some tasks, although some may argue that the scores produced by ACE are less intuitive than and not compatible with the more established measures of precision and recall. The scorer modules in GATE and Alembic are restricted for use within these two architectures and are more difficult to adapt for generic system evaluation.

With the above considerations in mind, we have designed ANNALIST (ANNotation ALIgnment and Scoring Tool), an evaluation system that is easily extensible and configurable for different domains, annotation tasks and input formats for both system developers and system users. ANNALIST offers a variety of options that allow it to be used either as a 'black box' system via the input of annotations in the form of XML or as a 'glass box' system via the use of an internal API.

The different sections of this paper are structured as follows: section 2 gives an overview of the software architecture of the ANNALIST system and section 3 describes the system's internal data representation; section 4 explains the criteria used for the comparison between annotations and section 5 discusses the algorithms for annotation alignment; section 6 describes ANNALIST's output in terms of scoring and alignment reports and section 7 discusses ANNALIST's results for a particular test collection in relation to results expected by other scorers (MUC).

2. ANNALIST Design

2.1 Scorer requirements

Those wishing to evaluate semantic annotation systems are usually one of two types:

Annotation system developers: These are typically programmers or software system developers and it is reasonable to assume that they may be comfortable working with Application Programming Interfaces (APIs). Such as a user might find it frustrating to work with a black box scoring system such as the MUC scorer (Douthat, 1998) that requires its input (i.e. the output of the annotation system) to be coerced into a standard format for

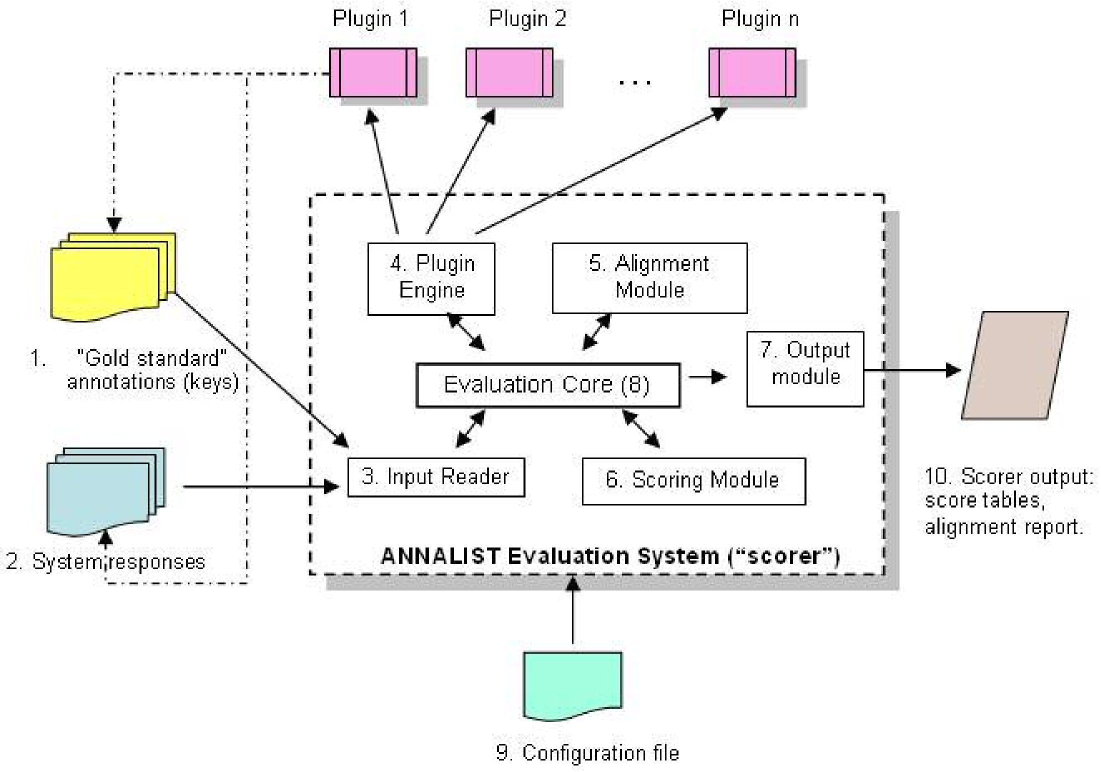


Fig. 1: Graphical representation of ANNALIST's architecture.

its data input and which offers limited options for configurability; instead, he/she might like to work with a highly modular or decomposable system where each system component, library or class is easily accessible through an API.

Annotation system users: These typically want to use the system in research or commercial applications. These users may include experts in their own fields, such as linguists or biologists, who may need to produce evaluation data but may be not have programming or system developing skills necessarily. Thus, it is safe to assume that such users may be more comfortable working with data files rather than APIs, perhaps generating their evaluation data using a text editor or XML tool.

Thus a first requirement for the design of ANNALIST is to build a system that allows both API access to core components of the architecture (input file parsers, aligner, scorer, etc.) and also allows running the system as a blackbox, albeit a configurable one.

A second requirement is the need for easy adaptation to different file formats, enabling the users integrate their own document processors or parsers with the scorer's main functions.

Finally, experience with a variety of evaluation tasks in information extraction projects has emphasized the need for flexibility in the criteria for evaluating certain types of annotations such as entities or relations. For instance, the evaluation of entities à la MUC (named entity task) has always assumed that positional information for an entity is available in the form of byte-offsets or other kind of indices and this information is used during annotation matching and alignment. However, there may be situations when such positional information is either limited (i.e. it may indicate just the document section the entity was found in), absent or simply not reliable for processing. ANNALIST caters for such cases by providing configuration options for the pairing and alignment of annotations and a flexible object representation that can be used for specifying the role of positional information (see section 3).

2.2 System Architecture

The system architecture of ANNALIST is illustrated in Fig. 1.

Input is provided in the form of document collections: (1) a collection used for the reference data usually created by domain experts which is sometimes referred to as the *gold-standard* or answer *keys*, and (2) a document collection with the annotations produced by the annotation system which are to be scored against (1), usually referred to as system *responses* (2). These corpora are processed by an Input Reader (3) module that reads their format and produces data representations compatible with the scorer's specification. Two input formats are currently supported as standard, the MUC format (an SGML variant) and the

XML annotation specification in the Semantic Web VIKEF project[1].

The main drawback to using static document collections of keys and responses is that when the annotation format changes, it has to be converted to the file format supported by the scorer in order to be "parsable". This may not be always desirable or even possible in certain situations. To enable the users extend the capabilities of the scorer for different formats and tasks, an architecture based on plugins has been developed. The Plugin Engine (4) in Fig. 1 has the task of dynamically loading a plugin developed by the user. The purpose of the plugin is similar to that of the Input Reader module i.e. to read the user's keys and responses files and convert them to data objects compatible with ANNALIST's internal representation for the annotations without having to access the document collections directly.

Information about the document collections of keys and responses to be parsed by the specified is specified in a configuration file (9). To work with the scorer, the plugin needs to be implemented in accordance to the scorer's API and be placed in the system's directory structure.

After data input and before scoring the annotations, the Alignment Module (5) is accessed to provide mappings between key and response annotations. The Scoring module (6) takes input from the Alignment module (5) and produces scores according to the chosen metrics for evaluation, i.e. *precision*, *recall* and *F-measure*. Although it can be argued that there can be no perfect measure for the evaluation of text mining systems (for example, see Luo (2005) for likely problems with the F-measure in relation to coreference evaluation), these measures have been widely adopted since the MUC competitions and this fact makes suitable for cross-system comparisons.

This data is finally passed to and formatted by the Output module (7) which is used for dumping the results (10). The goal of the Evaluation Core module (8) is to provide an API for the internal representation of annotations, and to coordinate or synchronise the flow of data from plugins and between the other software modules in the architecture.

ANNALIST has been developed as an open-source project as an open source intended for distribution via SourceForge (www.sourceforge.net).

3. Data Representation

The data representation of the scorer was largely developed for the evaluation tasks of scoring recognition of entity mentions and detection of relations between entity mentions.

[1] www.vikef.net

For the purposes of this evaluation system, an *annotation* is a structural unit that comprises a set of *attribute-value* pairs. Attributes can be *predefined* or *user supplied*. User supplied attributes can be unlimited in number and are optional. Predefined attributes, which must have their values set either via the input of data files or via the API, include the annotation *type* (e.g. "entity", "relation", etc.), annotation *name* (e.g. "person", "organization", etc.), *offset* information (e.g. "34-39", "paragraph_1"), the *value* (e.g. "B3H 3J5 Halifax Canada") and an *identifier* that is unique for each annotation and which takes natural number values as strings (e.g. "35"). For entities, the identifier has a dual role: to differentiate between entities and to point to the relative position of entity mentions in the document. When offset information is not available the entity identifiers are used by the scorer to align the entity annotations by order of occurrence.

ANNALIST's generic object representation allows for the realization and scoring of a wide range of annotation types by varying the settings of predefined and user supplied attributes and using a combination of different criteria for scoring the annotations.

4. Criteria for annotation comparisons

The criteria for comparison and scoring of annotation attributes offered by ANNALIST are based on practical case scenarios for the evaluation tasks in the VIKEF and CLEF[2] projects. During the pairing and alignment of annotations, an annotation can be classified as correct, partially correct, incorrect, missing or spurious. The overall score between a key and response annotation is the F-measure score between their corresponding attributes.

During entity comparisons, partial matches can occur not only due to system's inability to fully recognise an entity mention but also because the annotation system may have annotated a single actual mention as two or more distinct annotations. For this reason, ANNALIST's matching criteria for entities include options for *strict* matching (only 100% match is allowed) or *inclusiveness* which allows for partial (substring) matching. For instance, consider the comparison between the following annotations[3]:

Key:
<location>B3H 3J5 Halifax Canada</location>

and

Response:
<location>B3H 3J5 Halifax</location>
<location>Canada</location>

[2] http://www.clinical-escience.org/
[3] This is an example from a real annotation system.

In this example, both response annotations could be partially matched against the single key annotation. ANNALIST provides options to specify how the first match (i.e. “B3H 3J5 Halifax”) and a subsequent partial match (i.e. “<location>Canada</location>”) will be scored (correct, partially correct, incorrect or spurious). The user also has options for specifying the matching options in the reverse situation i.e. when partial matches occur between multiple key annotations and a single response annotation. This differs from the mechanism used in the MUC scorer where the partial string comparison is based on the removal of a list of premodifiers, postmodifiers and corporate designators from both the key and response annotations.

Alignment for relations in ANNALIST presupposes the alignment of entity objects pointed to by the relation attributes of the annotations. Kehler et al (2001) discuss the fact that the lenient strategy for template alignment used in MUC (which is based on the pairing and scoring of template slots) can produce optimistic results that overestimate the annotation system’s performance. ANNALIST provides options for both lenient and strict criteria for relation alignment. The lenient criterion specifies that any relation attribute can substantiate the case for a key and a response annotation to be candidates for alignment. In contrast, the strict criterion imposes the restriction that only when all relation attributes in the key and response match with each other will the two annotations be considered candidates for alignment.

A different set of criteria allows users or annotation system developers to get useful insights into the performance of the relation algorithm/component independently from the system performance for entity recognition and alignment. Such criteria introduce more transparency in the results of relation scoring especially for cases where the relations may point to missing (in the case of key) or spurious (in the case of response) annotations and can help the users understand the reasons for the possible under-generation of relations. ANNALIST deals with such cases by allowing relation annotations not to take part in the scoring (i.e. to be ‘unscored’). For example, consider the following cases of key and response annotations:

Key:

<Entity-1>:
Name: author
Id: 27
Mention: “X”

<Relation-1>:
Name: authorOf
Id: 185
Arg1: <27>
Arg2: <35>

<Entity-2>:
Name: document
Id: 35
Mention: “Y”

Response:

<Entity-1>:
Name: author
Id: 31
Mention: “X”

Because the entity pointed to by the attribute “Arg2” in the key relation <Relation-1> is missing from the response (and therefore no response relation was generated), the key relation <Relation-1> does not take part in the scoring. Information about the missing attribute(s) is recorded separately for user assessment at a later stage. This is different from the practice used in MUC where any such relation would be considered missing.

Similarly, when the entity pointed to by one of the relation attributes has been found to be incorrect, both key and response relations remain unscored. In contrast, MUC would score the response relation as correct even though the misaligned attribute would be scored as incorrect.

5. Annotation alignment and scoring

The main purpose of the alignment process is the generation of mappings between key and response annotations.

Entity alignment between the key and response annotations can be performed in one of three levels: *byte-offset level*, *document section level* or *order of occurrence* in the text. Entities are aligned at byte offset level when they occur within the same text span. This criterion represents a strong constraint and leads to more accurate alignment of entities.

Alignment at document section level is limited to annotations found within the same structural unit such as a sentence, paragraph or whole document. This is useful when explicit byte-offset information cannot be used for the comparison between key and response annotations but there is information (in the form of attribute value) that indicates which entities are found within the same structural unit of the text.

When no positional information about entities is available, the entities will be aligned according to their order of occurrence as specified by the identifier attribute.

Two alignment algorithms have been implemented: (1) a greedy search algorithm based on the MUC style of alignment as described in Douthat (1998), and (2) an alignment algorithm based on dynamic programming.

In the MUC-style alignment, the algorithm combines all key and response annotations pairwise in accordance to the chosen criterion (i.e. *byte-offset level*, *document section level* or *order of occurrence*) and produces alignment scores for each pair based on the F-score of their attributes. The list of key-response pairs is sorted by score and is traversed from the top. A key annotation is aligned with a response annotation under the condition that both of them are still not aligned. A key or response annotation that remains unaligned at the end of all the traversals will be considered missing or spurious respectively.

This heuristic type of search has been found to be quite accurate when alignment is performed at byte-offset level. However, when alignment is done at section level or order of occurrence, this heuristic kind of search may not be globally optimal as there is no guarantee that the local F-score computed for a key-response pair in the list optimises the overall score for all aligned pairs.

ANNALIST includes a second option for annotation alignment based on Dynamic Programming (DP). DP has been widely used for applications such as biological sequence matching or dynamic time warping in voice recognition. The basic idea is to build an optimal alignment between two data sets using the already established alignments of smaller subsets of these sets. For example, consider the following sets of annotations:

Key:

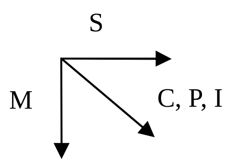
<person>John McCain</person> said <date>Tuesday</date> that he and <person>Barack Obama</person> were "moving on" after having a "nice discussion." In a letter to Sen.<person>Obama</person> on <date>Monday</date>,<person>McCain</person> -- upset by his colleague's support …

Response:

<person>John McCain</person> said <date>Tuesday</date> that he and Barack <person>Obama</person> were "moving on" after having a "nice discussion." In a letter to Sen.Obama on <date>Monday</date>, <person>McCain</person> -- upset by his colleague's support …

If no byte-offset information is taken into account, the MUC-style alignment algorithm would align the key person annotation "<Obama>" with the response annotation "Obama" because the F-score between them is higher than the score between the key "<Barack Obama>" and the response "<Obama>". The end result would be one correct and one missing annotation but this would be in disagreement with the true score as the correct alignment would generate one partially correct annotation in the response ("Obama") and one missing annotation ("Obama") in the key. Although such problematic cases may be rare in actual evaluation tasks, the potential for them to exist is real and the score differences may get large especially if alignment errors cascade in large annotation sets.

We use a variation of the DP algorithm by Gotoh (1982) to align the key with the response annotations. The algorithm is applied with two constraints: (a) the key and response annotations should be of the same type e.g. "entity" etc., and (b) the annotations should refer to the same entity name e.g. "person".



Response ⇒ / *Key* ↓	"John McCain"	"Obama"	"McCain"
"John McCain"	C		
"Barack Obama"		P	
"Obama"		M	
"McCain"			C

Fig. 2: Illustrated example of DP alignment. Alignment path is shown by the filled boxes. C = correct, P = partially correct, I = incorrect, M = missing, S = spurious.

A matrix *S(mxn)* with *m* and *n* the indexes for each set *Key* and *Response* respectively is built recursively with the value at each cell *S(i,j)* of the matrix being the overall score of the best alignment between the subsets $Key_{1...i}$ and $Response_{1...j}$.. By traversing the paths from points *(i-1, j-1)*, *(i, j-1)* and *(i-1,j)* to *(i,j)* the best score for *S(i,j)* is computed as the maximum of three options: (i) the score at *S(i-1,j)* minus a penalty *P* for aligning Key_i to a gap (i.e. a spurious response), (ii) the score at *S(i,j-1)* minus *P* for aligning $Response_j$ to a gap (i.e. a missing key), or (iii) the score at *S(i-1,j-1)* plus a score M computed by comparing Key_i with $Response_j$ (correct, partially correct or incorrect). An example of an alignment path found by DP is shown in Fig. 2[4].

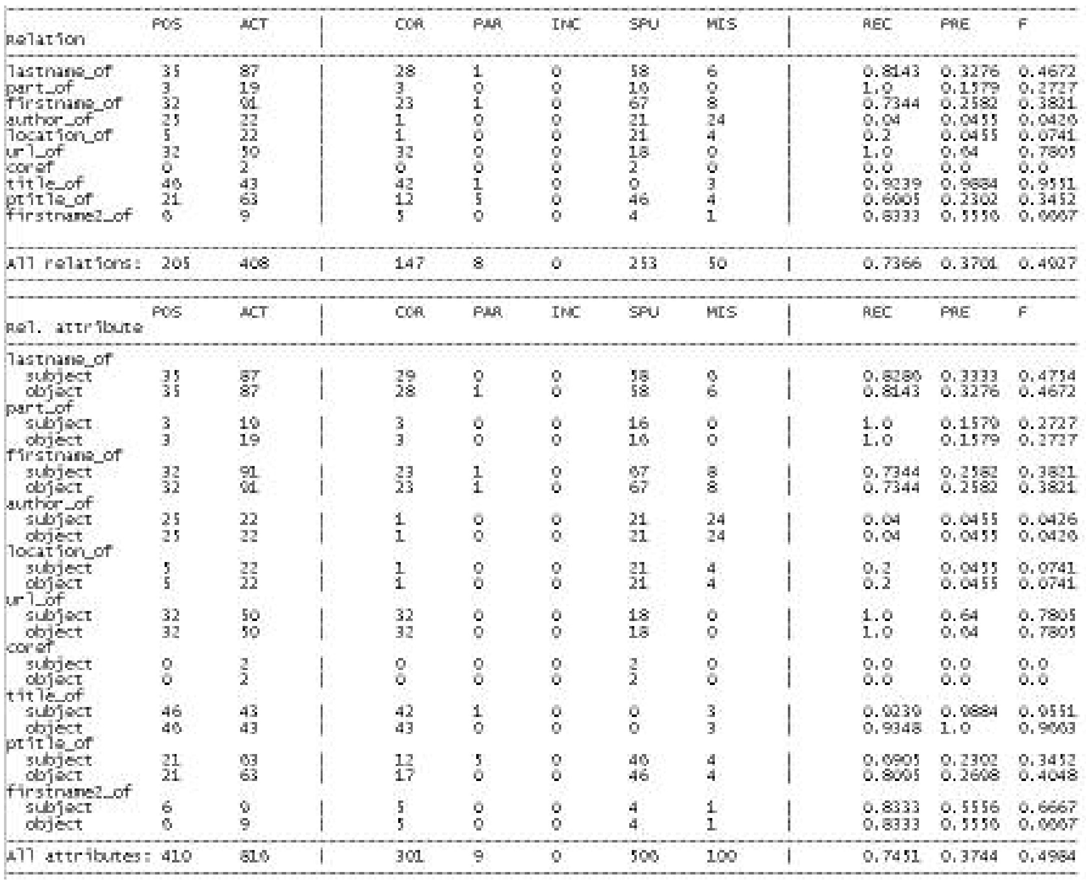
A restriction in the use of the DP-based alignment algorithm which does not apply for the MUC-style algorithm is that it requires annotations with their order of occurrence in the text preserved.

6. Evaluation reports

ANNALIST produces two kinds of reports: a report of the scores for individual documents as well as the collection as a whole (*scores report*) and a report that provides alignment and scoring details of all annotations during the evaluation (*alignment report*).

The scores report provides the document scores in three sections, one for entities, one for relations and one for the

[4] For the sake of simplicity in this example, we assume single-mention entities.



```
-----------------------------------------------------------------------------------------------------
                 POS      ACT    |        COR     PAR     INC     SPU     MIS    |        REC     PRE     F
Relation                         |                                               |
-----------------------------------------------------------------------------------------------------
lastname_of      35       87     |        28      1       0       58      6      |        0.8143  0.3276  0.4672
part_of          3        19     |        3       0       0       16      0      |        1.0     0.1579  0.2727
firstname_of     32       91     |        23      1       0       67      8      |        0.7344  0.2582  0.3821
author_of        25       22     |        1       0       0       21      24     |        0.04    0.0455  0.0426
location_of      5        22     |        1       0       0       21      4      |        0.2     0.0455  0.0741
url_of           32       50     |        32      0       0       18      0      |        1.0     0.64    0.7805
coref            0        2      |        0       0       0       2       0      |        0.0     0.0     0.0
title_of         46       43     |        42      1       0       0       3      |        0.9239  0.9884  0.9551
ptitle_of        21       63     |        12      5       0       46      4      |        0.6905  0.2302  0.3452
firstname2_of    6        9      |        5       0       0       4       1      |        0.8333  0.5556  0.6667

-----------------------------------------------------------------------------------------------------
All relations:   205      408    |        147     8       0       253     50     |        0.7366  0.3701  0.4927
-----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
                 POS      ACT    |        COR     PAR     INC     SPU     MIS    |        REC     PRE     F
Rel. attribute                   |                                               |
-----------------------------------------------------------------------------------------------------
lastname_of
  subject        35       87     |        29      0       0       58      6      |        0.8286  0.3333  0.4754
  object         35       87     |        28      1       0       58      6      |        0.8143  0.3276  0.4672
part_of
  subject        3        19     |        3       0       0       16      0      |        1.0     0.1579  0.2727
  object         3        19     |        3       0       0       16      0      |        1.0     0.1579  0.2727
firstname_of
  subject        32       91     |        23      1       0       67      8      |        0.7344  0.2582  0.3821
  object         32       91     |        23      1       0       67      8      |        0.7344  0.2582  0.3821
author_of
  subject        25       22     |        1       0       0       21      24     |        0.04    0.0455  0.0426
  object         25       22     |        1       0       0       21      24     |        0.04    0.0455  0.0426
location_of
  subject        5        22     |        1       0       0       21      4      |        0.2     0.0455  0.0741
  object         5        22     |        1       0       0       21      4      |        0.2     0.0455  0.0741
url_of
  subject        32       50     |        32      0       0       18      0      |        1.0     0.64    0.7805
  object         32       50     |        32      0       0       18      0      |        1.0     0.64    0.7805
coref
  subject        0        2      |        0       0       0       2       0      |        0.0     0.0     0.0
  object         0        2      |        0       0       0       2       0      |        0.0     0.0     0.0
title_of
  subject        46       43     |        42      1       0       0       3      |        0.9239  0.9884  0.9551
  object         46       43     |        43      0       0       0       3      |        0.9348  1.0     0.9663
ptitle_of
  subject        21       63     |        12      5       0       46      4      |        0.6905  0.2302  0.3452
  object         21       63     |        17      0       0       46      4      |        0.8095  0.2698  0.4048
firstname2_of
  subject        6        9      |        5       0       0       4       1      |        0.8333  0.5556  0.6667
  object         6        9      |        5       0       0       4       1      |        0.8333  0.5556  0.6667
-----------------------------------------------------------------------------------------------------
All attributes: 410       816    |        301     9       0       506     100    |        0.7451  0.3744  0.4984
-----------------------------------------------------------------------------------------------------
```

Fig. 3: A fragment of a scores report for the evaluation of relations.

relations' attributes. The scores are shown in tabular form and include figures for the correct, partially correct, incorrect, missing and spurious instances of each type as well as recall, precision and F-measure figures for all entities, relations and attributes. A small fragment of such a report is shown in Fig. 3.

The alignment report gives a full account of the alignment results between key and response annotations. For entities, all annotations that are correct, incorrect, missing or spurious are displayed with relevant information such as the annotation identifier, the entity mention which was used to generate the annotation, the start and end offsets in the key and response documents etc. For relations, information about the correctness of an annotation is accompanied with information about which relation attributes were found to be correct, incorrect, missing or spurious for this annotation. Finally, any 'unscored' relation annotations are reported in a separate part of the same report with extra information given about why the annotation was not scored i.e. which attribute was judged to be missing/incorrect in the annotation.

The added-value of the alignment report is that the user can easily apply simple text processing commands to find answers to questions such as "which entities are spurious", or "how many relations were unscored due to missing arg1" etc. and collect statistics on system performance.

7. Comparison with other scorers

ANNALIST has been used for evaluation exercises within the VIKEF and CLEF projects. For CLEF, ANNALIST was used for the evaluation of inter-annotator agreement exercise during the production of gold-standard annotations of CLEF's medical reports (13 annotators in total). For VIKEF, ANNALIST was used to evaluate VIKEF's semantic annotation performance on scientific articles.

In this section we report ANNALIST results from the VIKEF evaluation exercise that highlight some of the differences in the scores produced by this scorer with respect to scores which would have been expected from MUC.

The evaluation collection included 50 medical abstracts about Parkinson's disease from which a set of 7 entity types (document, title, person, organisation, location, date, url) and 11 relation types (firstname_of, firstname2_of, lastname_of, title_of, ptitle_of, author_of, member_of, location_of, part_of, url_of, coref) were extracted by VIKEF's annotation system and was scored against a human generated gold standard. To compare the ANNALIST's evaluation results with those expected from MUC we used ANNALIST with and without MUC string matching options.

On entity evaluation, MUC's strict criterion for partial string matching, which is based on the removal of premodifiers, postmodifiers and corporate designators from annotation stringfills, was found to be quite inflexible in practice as there is no fixed list of tokens that can be used to cover all cases. For example, when comparing the **key**: "Jovin, T. M." with the **response**: "Jovin T.", the initial "M." should be specified as postmodifier in MUC in order to account for partial matching. However, specifying such tokens for removal during string matching increases the risk that two unrelated entities (e.g. persons with the same last name but different first names) match with each other. On the other hand, if no removal of tokens is specified in MUC no partial credit is given. In practical terms, it is quite hard to make an accurate assessment of the differences in the scores expected by ANNALIST and MUC based on their options for partial matching as the user would be required to constantly update the MUC token lists and search for possible conflicts in order to simulate the ANNALIST's option for partial matches. However, it was found experimentally that ANNALIST's partial matching option[5] can result in an increase of about 3% in both precision and recall scores for this collection.

On relation evaluation, a comparison was made between the results obtained with the strict alignment criterion used in ANNALIST (that requires all relation attributes to match in order a key and a response relation to be candidates for alignment) and the results generated with the more lenient criterion used in MUC. It was found that the lenient alignment criterion generates somewhat optimistic precision and recall scores when compared to the corresponding scores produced with the strict criterion. Precision goes up from 77% (strict criterion) to 80% (lenient criterion) and recall goes up from 60% (strict criterion) to 62% (lenient criterion).

8. Conclusions and future work

This paper described ANNALIST, a new evaluation tool for the alignment and scoring of semantic annotations of text. Compared to other, more established evaluations tools, such as the MUC scorer, ANNALIST offers some advantages:

- A plugin facility for seamless integration of ANNALIST's alignment and scoring mechanisms with external modules that can be used to load the annotation data dynamically from document collections.

- Configuration options for the better treatment of partial matching between entities and strict criteria for relation matching.

- More transparency for the evaluation of relations by specifying criteria that can be used for excluding annotations from scoring.

- A new DP-based alignment algorithm for entity evaluation.

Tools like the MUC and ACE scorers still include a range of facilities that are missing from ANNALIST, such as the evaluation of coreference and scenario templates[6] and the specification of weights for entity and relation types amongst others so that ANNALIST should be considered as complementary tool rather than as a complete replacement for these scorers.

Future work includes the development of a graphical user interface for visual access to ANNALIST's plugin facility and the results of evaluation. Features of this interface include the setting of configuration options and the easy comparison of alignment and scoring results on screen.

Finally, we aim to extend the DP-style alignment algorithm in order to cover the evaluation of relations and types of annotation. We also aim to identify possible cases where ANNALIST's DP-based alignment algorithm differs from the MUC-style in order to gain more insight into the working differences between the two.

9. Acknowledgements

This work was supported by the VIKEF (EU FP6 - IST-2002-507173) and CLEF (UK MRC - G0100852) projects. We would like to thank our partners in these projects for their comments and suggestions during the development of ANNALIST.

10. References

ACE - Automatic Content Extraction (2007). At http://www.nist.gov/speech/tests/ace/index.htm.

Chatley R., Eisenbach S., Kramer J., Magee J. and Uchitel S. (2004). Predictable Dynamic Plugin Systems. In Fundamental Approaches to Software Engineering, *Lecture Notes in Computer Science*, vol. 2984/2004, pp. 129--143, Springer Berlin / Heidelberg.

Cunningham H., Maynard D., Bontcheva K., Tablan V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, pp. 168--175, Philadelphia, U.S.A.

Day D., Aberdeen J., Hirschman L., Kozierok R., Robinson P. and Vilain M. (1997). Mixed-Initiative Development of Language Processing Systems. *Proceedings of 5th Conference on Applied Natural Language Processing*, 31 March -3 April, Washington D.C., U. S. A, pp. 348--355.

[5] A perfect configuration of premodifiers, postmodifiers and corporate designators would be required to simulate the same functionality in MUC.

[6] ANNALIST's object representation can be configured to allow evaluation in such cases.

Douthat A. (1998). The Message Understanding Conference Scoring Software User's Manual. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*, Fairfax, Virginia, U.S.A.

Gotoh O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162, pp. 705--708.

Kehler A., Bear J., Appelt D. E. (2001). The Need for Accurate Alignment in Natural Language System Evaluation. *Computational Linguistics*, vol. 27, no. 2, pp. 231-248.

Luo X. (2005). On coreference Resolution Performance Metrics. Proceedings of Human Language Technology Conference and Conference on Empirical Methods in NLP, (HLT/EMNLP), pp 25--32, Vancouver, Canada.

Message Understanding Conference (1995). *Proceedings of Sixth Message Understanding Conference (MUC-6)* 6-8 November, Columbia, MD, Morgan Kaufmann.