Benchmarking Textual Annotation Tools for the Semantic Web

Diana Maynard

Dept of Computer Science, University of Sheffield, Sheffield, UK {diana}@dcs.shef.ac.uk

Abstract

This paper investigates the state of the art in automatic textual annotation tools, and examines the extent to which they are ready for use in the real world. We define some benchmarking criteria for measuring the usability of annotation tools, and examine those factors which are particularly important for a real user to be able to determine which is the most suitable tool for their use. We discuss factors such as usability, accessibility, interoperability and scalability, and evaluate a set of annotation tools according to these factors. Finally, we draw some conclusions about the current state of research in annotation and make some suggestions for the future.

1. Introduction

This paper investigates methods and results for benchmarking textual annotation tools. We define first some criteria for benchmarking, and examine those factors which are particularly important for a user to be able to determine which is the most suitable tool for their use. We then perform a series of experiments on a set of annotation tools, and discuss the results, finally drawing some conclusions about the future of annotation tools.

The term ”annotation” can be used to mean many different things: for example, simply adding comments to a document; adding information about document structure or composition, author, document type etc; adding linguistic information such as part-of-speech tagging, and so on. Here we refer to textual annotation as the process (or result) generally performed by means of some kind of ontology-based information extraction (OBIE). This consists of identifying the key terms in the text (such as named entities and technical terms) and then relating them to concepts in the ontology. In this paper we are concerned with tools that perform automatic rather than manual annotation.

There has been some previous work aimed at comparing and/or evaluating different annotation tools. However, this largely examines only the performance of the information extraction component and is based on a single set of task-specific data (see for example (Sazedj and Pinto, 2005)). Their aim is simply to examine the state of the art in annotation performance, paying little heed to the demands of real users of semantic web technologies. What such previous work ignores is that there are many more aspects to evaluating a tool than just measuring its performance in a particular situation. In our view, a complementary comparative study is required to look into issues going beyond performance as measured by precision and recall in quantitative evaluations. In this paper we analyse issues relating to usability, accessibility, scalability and interoperability, and investigate the extent to which some current annotation tools are ready for the demands of users. This aims to enable users to draw some conclusions about which (if any) tools are best suited to their needs. We make no claims that any one tool is a priori better than any other, because it depends on many factors such as who will be using it, what they will be using it for, and which features they care about most (e.g. speed vs. usability).

2. Requirements

According to (Reidsma et al., 2005), there are 3 groups of annotation tool users: annotators, annotation consumers, and developers (of which there are two types: corpus developers and system developers). In reality these groups may not be so well defined, for example, corpus developers are very often the same people as the annotators. However we should be aware of the differing abilities of these groups of users. We would not necessarily expect annotators to have linguistic skills, let alone computer skills, although they will probably have domain knowledge, whereas system developers can be expected to have computer skills but not necessarily domain knowledge or linguistic skills. Corpus developers will generally have linguistic skills and probably domain knowledge, while annotation consumers will probably have domain knowledge but not necessarily linguistic or computing skills.

Table 1 shows the extent to which each kind of requirement is important for the different users. A requirement which is not applicable (e.g. correctness of performance is not applicable when manually annotating) is denoted by ”-”. A single ”+” indicates that a requirement may be important or is slightly important; ”++” indicates a greater importance; ”+++” would indicate that a requirement is absolutely essential.

One of the main problems with designing automatic annotation tools (common to many other software tools) is the tradeoff between generality and specificity. Even though annotation may appear to be a quite straightforward task with a clear objective, both the uses and users of annotation tools may differ widely, and unless the tool is designed for a specific clear purpose with a particular kind of user in mind, there will almost always be dissatisfaction somewhere. One of the most important criteria is therefore that the annotation tool should be flexible and easily adaptable and/or extendable to the user's needs. This could range from something as simple as being able to change the colour of the annotation, through to enabling a whole new kind of visual representation or even modality.

3. Annotation Tools

The textual annotation tools we have chosen to investigate are GATE (Cunningham et al., 2002), KIM (Popov et al., 2004), OntoMat (Handschuh et al., 2002), MnM (Motta et al., 2002) and Magpie (Domingue et al., 2004). These

User	Manual Annotator	Annotation Consumer	Corpus Developer	System Developer
Usability	++	+	++	+
Flexibility	+	+	++	++
Performance	-	++	+	+
Scalability	-	++	+	++
Interoperability	+	+	+	++

Table 1: User Requirements for annotation tools

have been chosen for a number of reasons, but mainly because they all perform in some way automatic annotation of textual data with respect to an ontology (although they are quite diverse); they are all XML-based, open source, readily available and do not require extensive training to use. These tools have all been previously evaluated to some extent regarding performance, but have not been directly compared with respect to other criteria such as those we present.

4. Evaluation Criteria

In this section, we describe and discuss the evaluation of the annotation tools previously described, in terms of criteria such as usability, interoperability and accessibility issues. We describe our own experiences with the tools, augmented by comments and insights via a small survey of actual users of the tools.

4.1. Interoperability issues

Interoperability is concerned with how well the tool interacts with other tools and systems. Annotation is a task that is often combined with other applications, such as browsing, search and retrieval, indexing, etc., so it is important that annotation tools can easily interact with other systems. This is best achieved by conformance to existing standards. Interoperability evaluation not only covers annotation format, but also issues such as:

- data format: what kinds of text format can be processed, e.g. xml, html, sgml, txt, etc.;
- annotation schemes: whether annotation schemes can be imported/exported from other tools;
- plugins: if it is possible to plug in other tools and applications;
- API: whether the tool provides some API to programmatically access it.

Under the topic of interoperability, we investigate factors such as which platforms and browsers the tool runs on, ontology formats possible and how easily the tool can be modified.

4.1.1. Platform

This factor considers which platforms the tool can be run on, according to the documentation and/or discussion with the providers. All tools worked with both Windows and Linux apart from KIM which only worked with Windows. GATE, OntoMat and MAGPIE also worked with Macs.

4.1.2. Browser

Here we consider which browsers the tool works with. Here MAGPIE was the only tool which worked with both IE and Firefox. KIM worked only with IE, while MnM and OntoMat have their own proprietary browser and GATE does not use a browser as such but imports the documents into its own interface. GATE can, however, be run via its API as a web service on any browser.

4.1.3. Browser variation

This question looked at if and how there are any differences when the tool is run with different browsers. This only therefore applied to MAGPIE as the other tools only work with a single browser. It was found that there are quite a few differences with different browsers: when running on Firefox the performance was slightly substandard and there were some unexpected happenings; there were also conflicts between browser and plugin commands on Firefox. It is expected that later versions of MAGPIE will not have this problem, however.

4.1.4. Ontology format

This looks at which ontology formats are compatible with the tool. All the tools are compatible with OWL except for MnM (which is an older tool and no longer supported) which uses DAML+OIL and RDF, GATE is also compatible with RDF. At the time of testing, MAGPIE used its own proprietary ontology format but its current version supports OWL.

We also investigated specifically the OWL interoperability of the tools, by testing the ability of the tools to import and export OWL files without data loss. In fact, only GATE and KIM (which use the same underlying ontology repository, OWLIM (Kiryakov, 2006)) could be benchmarked in this way, out of the annotation tools under investigation, as the others either did not support OWL or could not be tested for various reasons. Compared with other semantic web tools, GATE (and KIM) actually performed very well in this respect, only suffering from some minor data loss. In particular, it interacted very well with Protege, KAON, JENA and SWI-Prolog, which were also among the best performing tools in the experiment. More details of this experiment can be found in (García-Castro et al., 2007). We can conclude from this that there seems to be a benefit for a semantic annotation tool to make use of a semantic repository such as OWLIM, which assists interoperability with ontology editors, support for ontology formats, etc.

4.1.5. Data format

Here we look at what kinds of textual format can be processed by the tool. GATE and KIM both score highly here by being able to process many formats, including plain text, SGML, XML, HTML, RTF, Word, and PDF (actually they use the same functionalities for document format analysis). The other tools only really process texts that can be displayed in a browser, i.e. HTML, plain text, XML etc.

4.1.6. Source available

This question looks at whether the source code is freely available so that developers can extend or modify the tool as required, for example adding new annotation sets, new visualisation capabilities, new processing resources, etc. This is a very important part of the flexibility and extensibility of the tool, since as discussed in Section 2., if the type of user and tasks for which the tool is to be used remain unknown or unpredictable at the time of design, then the tool needs to be able to cater for such flexibility if it is to fulfil interoperability requirements and be widespread in its use. The only tools of those tested that, to our knowledge, have this capability are OntoMat and GATE. However, there may be extensibility advantages with tools such as KIM that have open APIs.

4.2. Usability

Under the topic of usability, we categorise the quite broad issues such as documentation, ease of setup and installation, aesthetics of design, and range of tasks possible. Some more specific issues concerning accessibility are categorised separately in Section 4.3..

4.2.1. Installation ease

First we look at how easy the tool is to install. While this is clearly related to the following question about the installation documentation, the two may be orthogonal as the installation documentation could be very poor but installation may still be very easy. Indeed we can see this from the results, for example MAGPIE was deemed very easy to install but the installation documentation was deemed poor. In fact, all tools were very easy to install except for MnM, which was easy for Windows but not for Linux.

4.2.2. Installation documentation

The installation documentation was evaluated separately from the general tool documentation or user guide. MAGPIE was the only tool to have a negative score for installation documentation, which correlates with the difficulty of installation. In the case of KIM, GATE and OntoMat, the quality of the installation documentation turned out not to be that important since the tools were easy to install anyway.

4.2.3. Documentation quality

We then turned our attention to the quality of the main documentation. MAGPIE and KIM had very good documentation, which was clear, comprehensive and easy to follow. MnM's documentation was clear but very basic, while KIM's documentation was not very clear. GATE's documentation was very comprehensive but quite confusing and the users found it hard to locate the relevant information because it was so detailed.

4.2.4. Documentation format

The documentation format is closely linked with the quality of the documentation, showing that the more modalities in the documentation, the higher the quality. We investigated 4 aspects of the document format: whether it had step-by-step instructions, images/screenshots, movies/demos or just simple textual instructions. MAGPIE was the only tool which had clear step-by-step instructions, and also had both images and movies. OntoMat and GATE both had textual instructions, images and movies, while KIM had textual instructions and images, and MnM just had textual instructions. Clearly the combination of all text, movies and images was the clearest for users, although step-by-step instructions were not necessarily an improvement on simple basic instructions (both MAGPIE and Ontomat scored highly on documentation quality although only MAGPIE had the step-by-step instructions). As with the installation instructions, however, a tool might be easy to use even though the instructions are poor.

4.2.5. Linked help

A final part of the instruction examination looked at whether there was a help facility available directly from the tool. This could be either just a link to the documentation (as in the case of GATE) or a specific help function similar to most proprietary programs. It is very useful to have direct access to help without having to revert back to the website or downloaded instructions somewhere in the user's file system. MAGPIE had a button linking back to the website, from which the user guide could be found, but no direct link to a help facility. KIM had no help facility, while OntoMat and GATE both have linked help available: GATE links directly to the user guide via a button on the menu.

4.2.6. Configuration

The configuration criterion looked at how easy the tool was to set up in the way that the user wanted. This does not include installation, but rather things like changing the appearance of the GUI to suit the user's needs, changing options such as whether the tool should save session on exit, layout of menus, different skins or "look and feel", altering the fonts, etc. It does not consider the actual presence or absence of such options, but is based on how easy it is to set up the tool according to the options given, i.e. how easy these options are to use. MAGPIE and OntoMat were deemed very easy to set up (possibly because not so many options were available), while KIM was deemed fairly easy. MnM was considered quite hard to set up because of unnecessary and confusing dialogues, while GATE was deemed quite hard simply because many options were available and it was not always clear what they did. It is also not obvious in GATE that some options (such as changing the colour of annotations) are even possible, without reading the user guide.

4.2.7. Aesthetics

This question looked at how pleasing the tool was generally for the user, in terms of appearance, attractiveness etc. MAGPIE, KIM and GATE all scored highly here, and were considered to be colourful and interesting. MnM and OntoMat were rated quite poorly, and considered to be dull and unappealing. The aesthetics of a tool seems to be quite highly linked with the overall goals of the tool: MnM was designed as a very simple tool for a fairly limited range of tasks, and therefore it seems that not much consideration was given to its attractiveness. GATE on the other hand is designed for a very broad range of users and applications, and since one of its objectives is to be used as widely as possible, much consideration has gone into the look and feel of the tool.

4.3. Accessibility

Software accessibility is essentially about making tools that are usable, easy to use and attractive to use for everyone (not just for people with disabilities) (Maynard et al., 2007). Some of the most important examples of accessibility problems stem from *inflexibility*. A well designed tool will have options to change the user's preferences regarding colours, layout, font sizes and styles, and so on, and the ability to save and restore latest sessions, etc.
Even though a user should be able to choose such options, the default options should also be well designed. For example, text should be in a mixture of upper and lower case, and colour schemes should incorporate contrasting colours in the same range. Icons should be clearly understandable, not just with alternative text on mouseover, but should also use clear symbols and be large enough to click on easily. Mouse alternatives should also be widely available, although in practice it is hard to annotate regions of text manually without using a mouse.
In general, GATE performed the best on the accessibility tasks, with KIM a close second. Again, this is largely because GATE is much more flexible than the other tools, allowing the user to set it up how they want and to change the look and feel according to their preference.

5. Scalability

One aspect of scaling up to the real world is to combat the problems of accuracy when systems are deployed in industrial applications, such as the enabling of public metadata-on-demand services. However, even in a closed domain with a tightly focused application, there are more mundane issues concerning the sheer volume of data, with respect to storage, processing speed and power. Scalability has been identified as a critical issue for the processing of large volumes of data, so that statistical IE algorithms can be designed and trained (since these require large amounts of annotated training data in order to be accurate). In this section, we discuss some research designed to investigate the feasibility of scaling annotation tools up to the demands of the real world, i.e. performing annotation on an industrial level with massive volumes of data and/or huge ontologies.
SWAN [1] was a recent experiment in scaling up automated metadata extraction for industrial strength Semantic Web application development, involving 3 systems: GATE, KIM and a focused crawler called SECO. Following on from this, the KIM Cluster Architecture has been developed with the specific aim of extensive scaling of the existing KIM model (Manov and Popov, 2005). The system has been demonstrated on a corpus of over 1.2 million news articles, resulting in the recognition of over 1 million named entities. These were all stored in the semantic repository based on OWLIM, along with their semantic descriptions (attributes and properties). The average speed of annotation in this experiment was 10KB per annotator node/component.

The scalability of GATE has been demonstrated through a number of experiments running various language processing tasks over large textual collections (Maynard et al., 2007). The machine learning API is capable of running on huge amounts of data, e.g. very large ontologies (dependent on the hard disk size) and thousands of documents for training (depending on memory size and – for some learning algorithms – hard disk size). This is mainly due to the datastore mechanism in GATE, which stores immediate data and results on the hard disk rather than in memory, loading and unloading documents as they are processed.

The other 3 annotation tools examined were not really designed for large-scale applications, and this is their biggest drawback when using such tools for industrial rather than testbed research applications. The main aim of MnM was to provide proof-of-concept rather than a serious industrial-strength tool. MnM has serious problems when used with even a medium-sized ontology, and since it relies on training, a user first has to manually annotate documents, which is infeasible on a large scale and with an ontology of any significant size.

One aspect of scalability that showed particularly strongly in the case of Magpie was the relationship between the desire to scale up as opposed to the real time responsiveness of the application to the user's interaction. Thus pragmatically, scale is only an added value of a particular annotation tool as far as the user is willing to wait for the outputs: for GATE, this is easily in the extent of several hours or even days, since the output is a marked-up collection of documents. For Magpie, the time limits are in the range of seconds – otherwise, the user is less likely to achieve the main objective of this tool: navigation on the Web using semantic relationships and links.

In summary, GATE and KIM are designed to be scalable and to be applicable in real world scenarios. Extensive experiments - both on the tools themselves and with their use in industrial settings - have shown that they can cope reasonably well with large volumes of data and real-life ontologies without sacrificing performance. There are, however, some limitations: GATE applications may be quite slow depending on both the approach used and the size and content of the texts. MnM was never designed to be used in a large-scale environment and therefore suffers from a number of problems (some of which are indeed insurmountable without modifying the actual tool) when applied to large ontologies. Similarly, OntoMat was never designed for scalability and can suffer problems when used in such an environment. In general, any system that relies on learning will have problems when applied to a new ontology, as it will require a large amount of manually annotated training data which is time-consuming to produce.

[1] http://old.deri.ie/projects/swan/

6. Recommendations and Conclusions

In this paper, we have introduced a set of different annotation tools and investigated various aspects such as their aims, usability and scalability. Our aim is not to try to identify which tool is the best, but rather to examine the features a user should look for when deciding on a tool to use, and to highlight some issues which should be factors in that user's choice.

It is clear from our research that the most important thing to consider is the task to which the tool will be put and the situation in which it will be used. For this, we must first examine two main factors: the type of user, and the scope of tasks on which the tool will be put to use. Once these have been identified, we can turn our attention to more details: for example, if the tool is to be used solely by annotators, then a high degree of usability straight from the box may be important; if the tool is to be used only for a single, standalone kind of task, interoperability may be less of an issue than if the tool is to be used within a larger suite of tools, and so on. In Section 2. we established a set of requirements and investigated the importance of each with respect to the kind of user. In Table 2 we summarise the extent to which the annotation tools we have analysed fulfil these requirements. As before, "-" indicates the absence or near absence of that requirement, while the number of + signs indicate the level. While it appears from the table that GATE is in general the best performing tool overall, it really depends what a user's requirements are. For example, if a simple or highly scalable tool is required, it may not be the best choice.

In this paper, we have not approached the subject of performance of the tools. This is for two reasons: (1) performance evaluations of annotation tools have been documented elsewhere, e.g. (Maynard et al., 2007; Sazedj and Pinto, 2005; Maynard et al., 2008), and in the Dot.Kom and SEKT projects; (2) previous evaluations have shown that it is difficult to compare different tools on the same documents and with the same ontology, since they are all designed for different purposes and work best on the kind of data they were designed for, so any comparison may lead to unfair prejudice. For example, in the case of Magpie, the precision on annotating text that is within the domain of the loaded lexicon is far superior to most other tools; however, the recall tends to decline rapidly if the text moves beyond the domain of the lexicon. This is a well-known feature of ontology-driven annotation, often presented as ontology brittleness (Dzbor and Motta, 2006). We refer the reader, however, to (Maynard et al., 2007) for a look at the performance of the tools in question in a number of experiments.

The problem of domain dependence is also related to the issue of algorithmic reuse. The IE systems underlying the annotation tools are largely tailor-made for specific domains and applications, with the result that not only are they hard to adapt for new tasks, but that it is difficult to extricate potentially reusable components or sub-components which are buried deep in the architecture. For example, there may not be a distinction between foreground information, which is dependent on the domain and application, and background information, which can be reused as it stands; and consequently, between the tools needed to access and manipulate these two types of information.

Annotation tools have already proved their success in a number of real world applications, such as Del.ici.ous, Flickr, digital libraries such as Perseus[2], Garlik[3] (which mines data about consumers present in various sources including the web), Fizzback[4] (which provides real-time customer feedback from SMS and email feeds) and so on. However, there are several reasons why semantic annotation is not more widespread. First, it is time-consuming and complex to produce annotations in open domains. Second, there are not enough DIY cases: Flickr and Del.ici.ous have tapped into obvious needs, and Innovantage hopes to do the same, but in many cases we do not have enough folk for a folksononmy. For example, broadcast archives are a Catch-22 situation: people are not aware that they need them until they use them, and they cannot use them until widespread need generates either human effort or funding. Automatic methods for annotation, on the other hand, have made huge advances in recent years to a usable level, but the output still tends to be incomplete or innacurate, especially in open domains. One solution is therefore to combine automatic methods with human annotation at the lowest possible cost, and preferably done by non-experts. While we have plenty of algorithms, data structures and evaluation protocols emerging, we still currently lack a clear statement of how to specify and implement new annotation tasks, especially those oriented towards non-HLT experts. What is therefore needed is a methodology covering how to:

- decide if annotation is applicable to a problem;
- define the problem with reference to a set of examples;
- identify similarities with other problems and thus to estimate likely performance levels;
- design the annotation workflow, including automatic assistance;
- measure success.

These tasks are beyond the scope of the current work; however, they are currently being pursued in projects such as NeOn[5] and MUSING[6], by researchers at Harvard Medical School, and by some commercial users.

Acknowledgements
The research for this paper was conducted as part of the EU Network of Excellence Knowledge Web.

7. References

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

[2]http://www.perseus.tufts.edu/
[3]http://www.garlik.com
[4]http://www.fizzback.com
[5]http://www.neon-project.org
[6]http://www.musing-project.eu

Tool	GATE	KIM	MnM	Magpie	OntoMat
Usability	++	++	-	++	+
Flexibility	++	+	-	+	+
Performance	+++	++	+	++	++
Scalability	++	+++	-	++	+
Interoperability	++	++	-	+	++

Table 2: Requirements fulfilled by annotation tools

J. Domingue, M. Dzbor, and E. Motta. 2004. Magpie: Supporting Browsing and Navigation on the Semantic Web. In N. Nunes and C. Rich, editors, *Proceedings ACM Conference on Intelligent User Interfaces (IUI)*, pages 191–197.

M. Dzbor and E. Motta. 2006. Study on Integrating Semantic Applications with Magpie. In *15th Intl. Conference on Artificial Intelligence: Methods, Systems & Applications (AIMSA)*, Bulgaria.

R. García-Castro, S. David, and J. Prieto-González. 2007. D1.2.2.1.2 benchmarking the interoperability of ontology development tools using owl as interchange language. Technical report, Knowledge Web, September.

S. Handschuh, S. Staab, and F. Ciravegna. 2002. S-CREAM — Semi-automatic CREAtion of Metadata. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 358–372, Siguenza, Spain.

Atanas Kiryakov. 2006. OWLIM: balancing between scalable repository and light-weight reasoner. In *Proc. of WWW2006*, Edinburgh, Scotland.

D. Manov and B. Popov. 2005. D2.6.1 Massive Automatic Annotation. Technical report, SEKT EU Project Deliverable.

D. Maynard, S. Dasiopolou, S. Costache, K. Eckert, H. Stuckenschmidt, M. Dzbor, and S Handschuh. 2007. Benchmarking of annotation tools. Technical Report D1.2.2.1.3, KnowledgeWeb Deliverable.

Diana Maynard, Yaoyong Li, and Wim Peters. 2008. Nlp techniques for term extraction and ontology population. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text*. IOS Press.

E. Motta, M. Vargas-Vera, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. 2002. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 379–391, Siguenza, Spain.

B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. 2004. KIM – A semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10:375–392.

D. Reidsma, N. Jovanovic, and D. Hofs. 2005. Designing annotation tools based on properties of annotation problems. In *Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research*, Wageningen, The Netherlands.

P. Sazedj and H. Sofia Pinto. 2005. Time to evaluate: Targeting annotation tools. In *Proceedings of ISWC 2005 SemAnnot Workshop*, Galway, Ireland.