QUICK PAD TAGGER: AN EFFICIENT GRAPHICAL USER INTERFACE FOR BUILDING ANNOTATED CORPORA WITH MULTIPLE ANNOTATION LAYERS

Marc Schreiber[1], Kai Barkschat[1], Bodo Kraft[1] and Albert Zündorf[2]

[1]FH Aachen, University of Applied Sciences, Germany, Jülich
{marc.schreiber,barkschat,kraft}@fh-aachen.de
[2]University of Kassel, Germany, Kassel
zuendorf@uni-kassel.de

ABSTRACT

More and more domain specific applications in the internet make use of Natural Language Processing (NLP) tools (e. g. Information Extraction systems). The output quality of these applications relies on the output quality of the used NLP tools. Often, the quality can be increased by annotating a domain specific corpus. However, annotating a corpus is a time consuming and exhaustive task. To reduce the annotation time we present a custom Graphical User Interface for different annotation layers.

KEYWORDS

Natural Language Processing, Language Resources, Annotated Corpora, Annotation Layers, Annotation Speed, Graphical User Interface

1. INTRODUCTION

Many modern applications are analyzing natural language resources. For example, companies process incoming mails automatically and they are trying to identify if customers are satisfied with the provided services or not. To facilitate such analyses these applications apply Natural Language Processing (NLP) techniques, deriving meaning from natural language. In conclusion, the quality of the analyses relies on the output quality of the NLP tools.

The output quality of NLP tools varies when the tools will be applied in different domains [1, 2]. When NLP tools are used in specific domains, the quality can be increased by developing a domain specific algorithms [3, 4, 5] or training existing NLP tools on domain specific corpora [6, 7]. In both cases an annotated corpus is required to evaluate the output quality of NLP tools in different domains.

The creation of an annotated corpus is a time-consuming and error-prone process. To support and improve the annotation process Graphical User Interface (GUI) tools have been evolved, supporting annotators to create annotated corpora [8, 9, 10]. Often these tools offer to annotate a specific annotation layer (e. g. Named Entities (NEs)) with a specific GUI for this layer. Other tools provide to annotate multiple arbitrary layers with a generic GUI for all annotation layers.

David C. Wyld et al. (Eds) : CCSIT, SIPP, AISC, NLP - 2015
pp. 131–143, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.50413

However, both types of annotation tools have their pros and cons. Annotation tools providing to annotate a specific layer are optimized to annotate this specific layer, but at the same time they are restricted in their functionality. In contrast to these tools, other annotation tools can annotate multiple layers but they do not provide an optimized GUI for any annotation layer, resulting in higher annotation effort.

In this paper we present the Quick Pad Tagger (QPT) which closes the gap between layer specific and multiple layer annotation tools. The QPT provides to annotate multiple annotation layers and for each layer the QPT provides an optimized GUI.

The constellation of multiple annotation layers and specific GUI leads to better annotation speed. Additionally, the QPT is designed as semi-automatic annotation tool, providing suggestions to the user. The semi-automatic manner of the QPT increases the annotation speed furthermore.
This paper is structured as follows: Section 2 will outline related work. Following that, Section 3 will introduce the QPT followed by an evaluation in Section 4. Finally, Section 5 gives a conclusion and outlines future work.

2. RELATED WORK

Many studies show that annotation time can be reduced when an annotated corpus will be developed. In the clinical domain Lingren et al. [11] demonstrate that pre-annotation reduces the annotation time significantly for the NE annotation layer. As an additional result they reviled that pre-annotation did not influence the Inter Annotator Agreement (IAA) or annotator performance.
Loftsson et al. [12] investigate to use pre-annotations for the Part-of-Speech (POS) tag annotation layer. The authors describe that using pre-annotations reduces the effort of developing an annotated corpus. Fort and Sagot [13] investigate pre-annotation methods for POS tags more deeply and they verify that those methods result in better annotation accuracy.

Lingren et al. [11], Loftsson et al. [12], and Fort and Sagot [13] investigate the annotation process with pre-annotated corpora and they show that annotation time can be reduced. We reuse the idea of pre-annotation in the semi-automatic annotation process of the QPT (c. f. Section 3.2), resulting in higher annotation speed (c. f. Section 4).

SALTO [8] is a specific GUI annotation tool. SALTO is able to annotate syntactic structures in TIGER XML corpora—Mengel and Lezius [14] describe the TIGER XML corpus format in detail. SALTO also enables to add semantic classes and roles to TIGER XML corpora. Both features are based on graph representations and use a mouse based input method.

Knowtator [9] is a generic annotation GUI tool, implemented as a Protégé plugin [15]. The annotation schema can be defined by Protégé's knowledge-based editor which enables the generic annotation manner. Knowtator's input method is a mouse based input method. Nevertheless, Knowtator is very difficult to use for unskilled annotators.

Webanno [10] is a web-based annotation tool. It also provides a generic user interface to annotate different annotation layers with a mouse based input method. The annotation configuration is hidden to the user which makes it easier to annotate documents. A new version of Webanno provides annotation suggestions to the user as well [16].

All those annotation tools [8, 9, 10] are supporting a single annotation layer or they provide a generic GUI for multiple annotation layers. However, in the first case the functionality is limited and in the second case the annotation process is not optimized regarding annotation time. The QPT closes this gap by providing a specific GUI for each annotation layer.

3. ANNOTATING CORPORA WITH THE QUICK PAD TAGGER

The Quick Pad Tagger (QPT) follows a general design principle which is embedded into its name: Annotating text corpora (refers to Tagger) should be done as quickly as possible (refers to Quick) and therefore a minimal set of input keys is used (refers to Pad/Gamepad which provides only a limited set of keys). Inspired by a video game input method, the motivation of using a keyboard based input method can be described by the following reasons:

1. **The input method should be efficient:** Both Lane et al [17] and Omanson et al. [18] show that keyboard based input methods are often more efficient than mouse based input methods.

2. **The input method prepares to embed gamification elements:** Annotating a corpus is a time-consuming and monotonous process. Gamification is an element to improve the user experience [19]. With an improved user experience the annotation process is less exhausting.

Additionally, the QPT is a semi-automatic annotation tool for multiple annotation layers. This is motivated by the fact that suggesting annotations can improve the annotation speed (c. f. Fort and Sagot [13] and Yiman et al. [16], more details will be provided by Section 3.2).

3.1. Graphical User Interface of the Quick Pad Tagger

Currently, the QPT supports to annotate the following layers:

- Text Segmentation (TS), c. f. Figure 1
- Part-of-Speech (POS), c. f. Figure 2
- Named Entities (NEs), c. f. Figure 3
- Constituency-based Parse Trees (CPTs), c. f. Figure 4

The user can switch between the annotation layers by using the corresponding buttons (c. f. top of Figure 1). The user can switch anytime among the layers, unless required information are provided. For example, the Constituency-based Parse Tree (CPT) annotation layer requires POS information. If the document does not contain any POS tags, the user is not able to select the CPT annotation layer.

Figure 1 illustrates adding Text Segmentation (TS) annotations to a document. To annotate the TS information the QPT provides a cursor based input method. Basically, the user moves the cursor with the arrow keys and marks end of sentences with enter (blue marks in Figure 1). For splitting words into multiple tokens the user can use the spacebar (green marks in Figure 1). If one of the annotations is incorrect, the user can remove the annotations with the delete key.

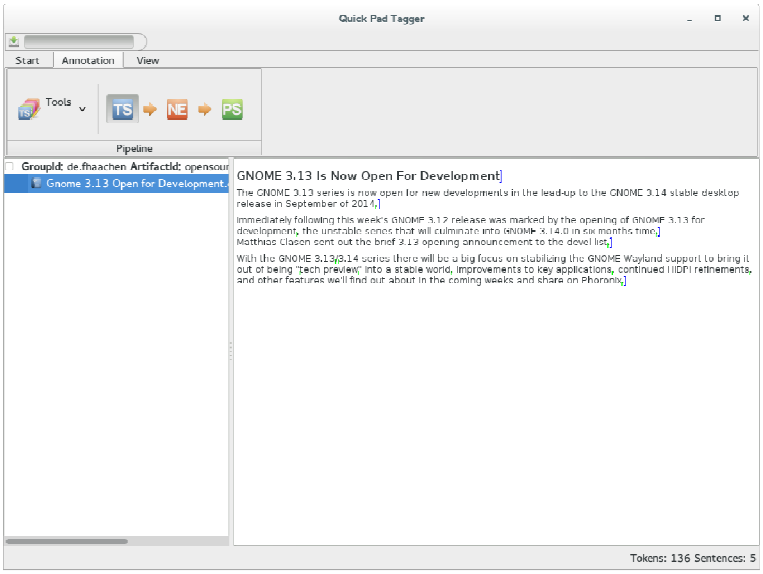

Figure 1: Text Segmentation Annotation with Quick Pad Tagger

To increase the annotation speed for the TS layer the cursor based input method moves from token to token because a character based moving slows down the annotation speed. The text displayed in Figure 1 contains of 746 characters. By using a character based input method the user would have to press the right arrow key 746 times to move the cursor from the beginning to the end of the document. After an initial whitespace tokenization the text contains of 124 tokens which reduces pressing the right arrow key to 124 times. To move the cursor from character to character the user just needs to press the control key.

Figure 2 displays the process of annotating POS tags. The QPT provides a sentence based selection and for each selected sentence the QPT shows a popup menu. By pressing left or right arrow key a token will be selected and by pressing up or down arrow key a POS tag will be assigned.

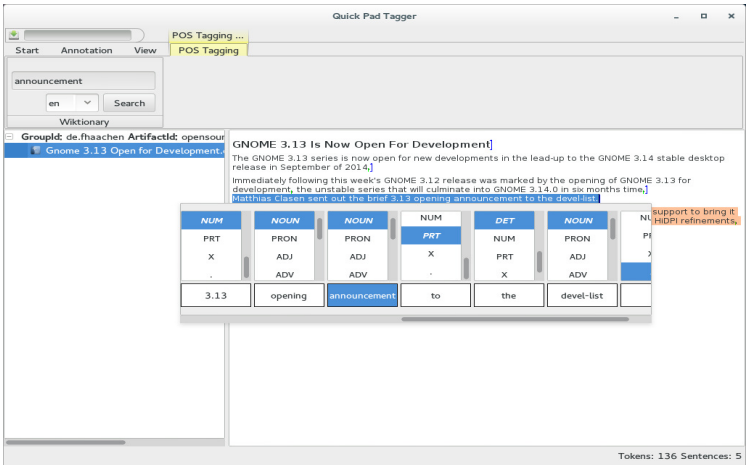

Figure 2: Part-of-Speech Tags Annotation with Quick Pad Tagger

Often annotators look up words in dictionaries like Wiktionary—Wiktionary also serves a platform for other NLP resources [20]. To reduce the effort to open Wiktionary and search for a specific word, the QPT provides an embedded search function (c. f. top left corner in Figure 2). When the user selects a token, the value of the token will be inserted into the text box. By pressing the F1 key the QPT will search on Wiktionary for the selected value which improves the annotation speed.

Figure 3 illustrates the process of annotating NEs. The QPT provides a token based selection which can be changed with the left and right arrow keys. The selection will be extended by pressing shift and arrow keys. The concepts of the NEs are displayed in a popup menu. The annotator selects another concept by pressing the up and down arrow key. By pressing the enter key the user assigns the selected concept to the selected tokens.

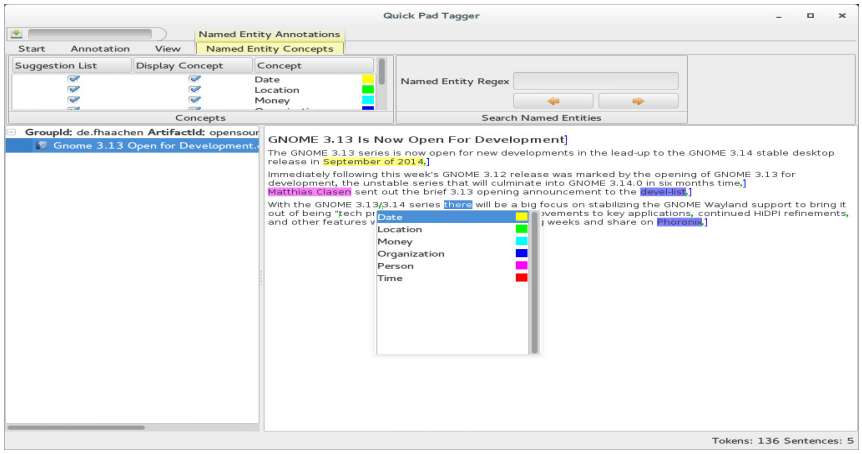

Figure 3: Named Entity Annotation with Quick Pad Tagger

Figure 4 shows the process of annotating CPTs. Similar to annotating POS tags the QPT provides a sentence based selection with an additional popup menu. The user can combine tokens to phrases and assigns each phrase a phrase tag. The left and right arrow selects a token or phrase and with the shift key the selection can be expanded. The spacebar is used to combine tokens or phrases to a new phrase. By pressing up or down arrow key the annotator selects a phrase tag.

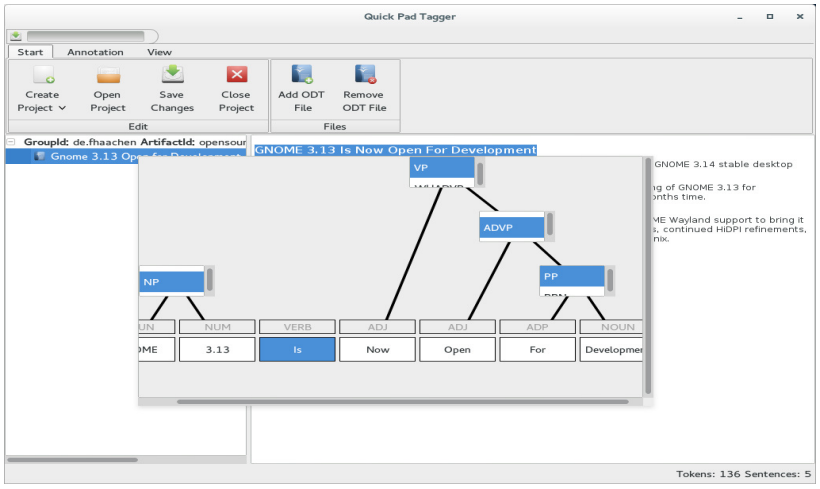

Figure 4: Constituency-based Parse Tree Annotation with Quick Pad Tagger

3.2. Semi-automatic Annotation Process

As mentioned at the beginning of Section 3 the QPT is a semi-automatic annotation tool. In this section we will provide an explanation how generating suggestions works (illustrated in Figure 5). When the first document will be added to the corpus, the annotator has to annotate the whole document without any suggestions. When the first document has been annotated, the QPT can access the previous annotation information for providing suggestions to the annotator.

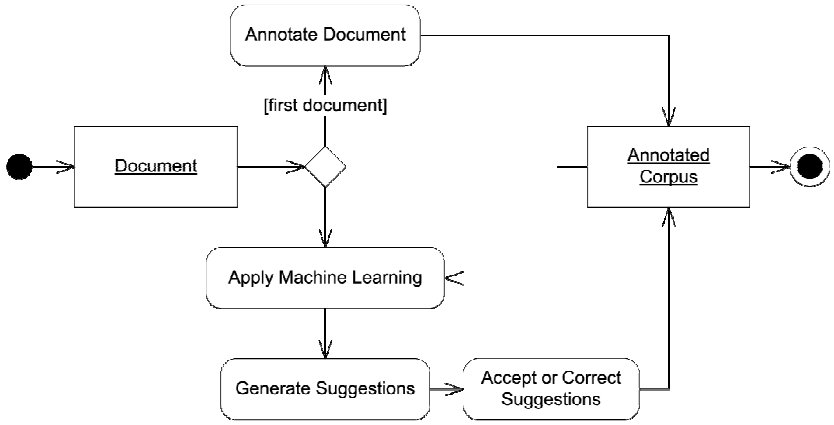

Figure 5: Semi-automatic Annotation Process

Therefore the QPT takes the information of the already existing annotated corpus and applies Machine Learning (ML) methods to generate a ML model. After that, the QPT uses the ML model to provide suggestions to the annotator. Finally, the annotator has to accept or correct the suggestions.

The ML methods for providing the annotation suggestion can be selected by the user. Through a plugin system the QPT can be extended with different NLP implementations—OpenNLP [21] and Stanford CoreNLP [22] are two example NLP implementations, providing annotation suggestions to the user.

4. EVALUATION

For the evaluation process of the conceived QPT GUI design we define the annotation speed metric as tokens per second for every annotation layer. For example, if a user is able to annotate a document consisting of 100 tokens with POS tags in 50 seconds, the annotation speed is two. This metric is independently of the used annotation tool and makes it possible to compare annotation tools regarding their effectiveness.

For the evaluation we compare the annotation speed of Webanno [10] with the QPT. OpenNLP provides the annotation suggestions for the QPT. Webanno uses a generic annotation GUI for every annotation layer and the QPT uses a specific GUI for every annotation layer. This setup was chosen to verify the hypothesis that a specific GUI for each annotation layer leads to higher annotation speed.

The annotation speed for the comparison has been measured by four different users annotating the same corpus with the following annotation layers: TS, NEs and POS tags. For annotating NEs we use the concepts Date, Location, Money, Organization and Person because these concepts are

known by the annotators. As POS tag set we use the universal POS tag set of Petrov et al. [23] because it is easy to learn for German texts.

Two users annotate the corpus with Webanno (Annotator A and B) and the other two users annotate the corpus with the QPT (Annotator C and D). None of the users annotated a corpus before. This setup of easy to understand annotation layers and non-experienced annotators ensures both: The annotation process does not take too long and none of the annotators is biased by a known annotation tool.

During the comparison we annotate a small corpus consisting of six documents. On average each document consists of 330 tokens. The documents come from different German online news portals:

1. Golem.de: `http://www.golem.de/`
2. stern.de GmbH: `http://www.stern.de/`
3. Süddeutscher Verlag: `http://www.sueddeutsche.de/`

The following three sections describe the evaluation comparing the annotation speed of Webanno and the QPT. Each section describes the results of the experiment comparing Webanno and the QPT for one annotation layer (TS, POS and NEs). Additionally, each section illustrates how the semi-automatic annotation process influences the annotation speed. Therefore, we analyze the F_1 score or accuracy of the suggestions: The F_1 score/accuracy is measured by taking the previous documents, generating the ML model, generating suggestions and comparing the users' annotations with the suggestions.

4.1. Text Segmentation Annotation Layer

Webanno provides no feature to annotate TS information. To be able to compare the QPT with Webanno we use following approach: The Annotators A and B create a plain text file with a text editor. Each line in the text file contains one sentence and the tokens are separated by whitespaces. The text files are then converted to Text Corpus Format files [24] and then imported into Webanno.

Table 1 shows the average annotation time for each document of the corpus. The QPT reduces the annotation time of TS information by 27 percent.

Table 1: Average Annotation Time for Annotating a Document with Text Segmentation

Webanno		**Quick Pad Tagger**	
Annotator A	Annotator B	Annotator C	Annotator D
2:28 min	2:25 min	1:45 min	1:50 min

Figure 6 displays the annotation speed for each document and annotator, annotating TS information. At the beginning the annotation speed of each annotator is almost the same. After annotating the first document the annotation speed of Annotator A and B increases slightly and stays more or less the same (on average 2.41 tokens per second). In contrast to the annotation speed of Annotator A and B the annotation speed of Annotator C and D increases constantly. At the end of the experiment the annotation speed of Annotator C and D is on average 2.3 times higher.

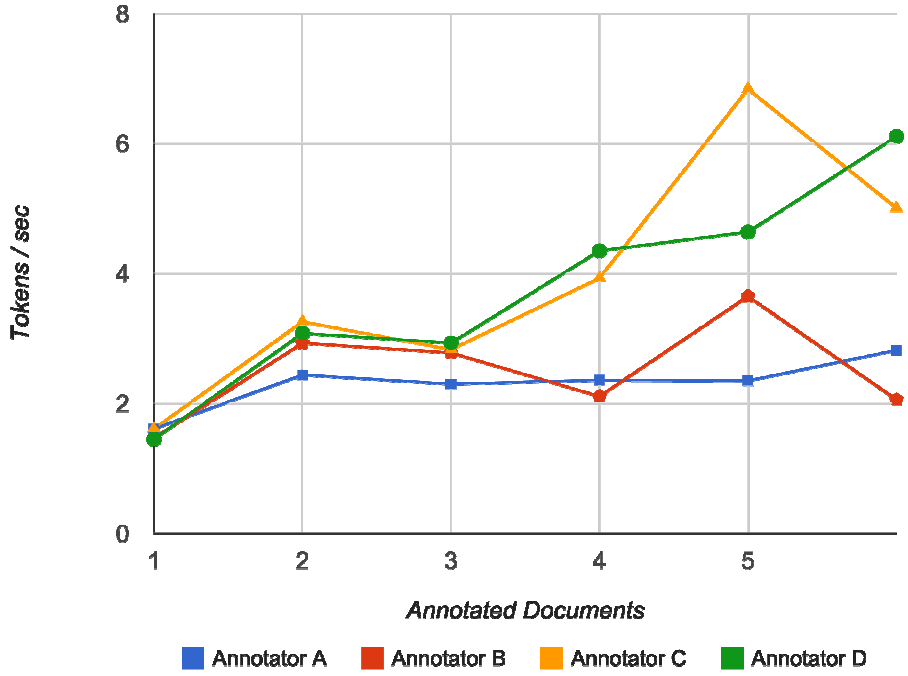

Figure 6: Annotation Speed for Text Segmentation

Figure 7 shows the F_1 score of the suggestions. On average the F_1 score is 93 percent. In conclusion, the annotation task of the Annotators C and D limits to reading the text and verifying if suggestions are correct. This limitation leads to a speedup compared to the Webanno approach.

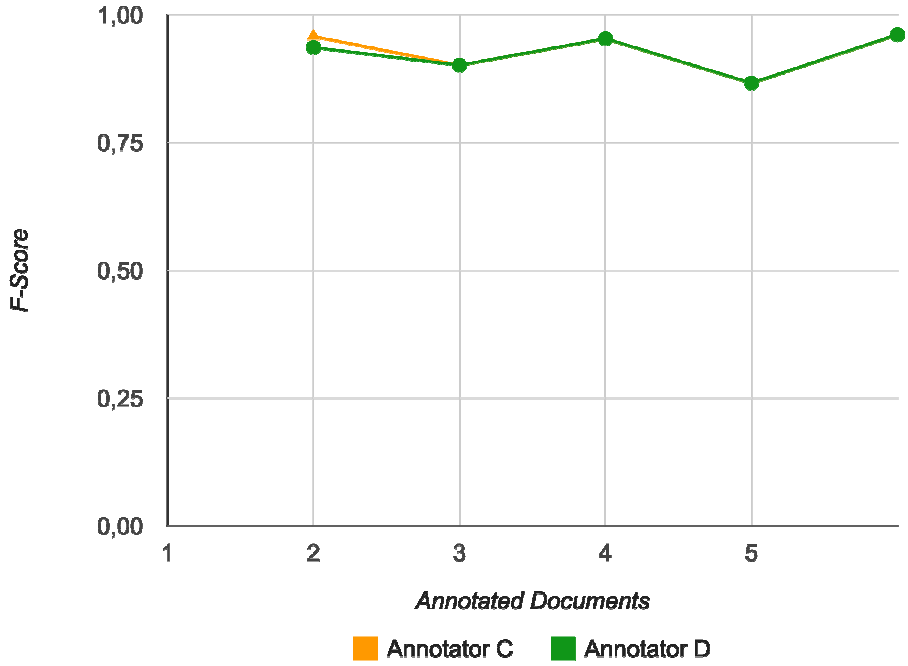

Figure 7: F_1 Score Regarding Suggestions for Text Segmentation

As the results of this experiment show, a custom GUI for annotating TS information is worthwhile. In our experiment the annotation speed could be improved by a factor of 2.3. Generating TS suggestions gives an additional speed up.

4.2. Part-of-Speech Annotation Layer

Table 2 shows the average time spent to annotate POS tags. The difference between using Webanno and the QPT are significantly: The QPT reduces the annotation time by a factor of 2.5 on average.

Table 2: Average Annotation Time for Annotating a Document with Part-of-Speech Tags

Webanno		**Quick Pad Tagger**	
Annotator A	Annotator B	Annotator C	Annotator D
42:19 min	51:15 min	16:59 min	20:49 min

Figure 8 displays the annotation speed for each document and annotator. On average the annotation speed is 0.2 tokens per second higher by using the QPT. Additionally, the annotation speed of Annotator C and D increases better than the annotation speed of Annotator A and B. At the end of the experiment Annotator C and D annotate faster by a factor of 2.5 on average.

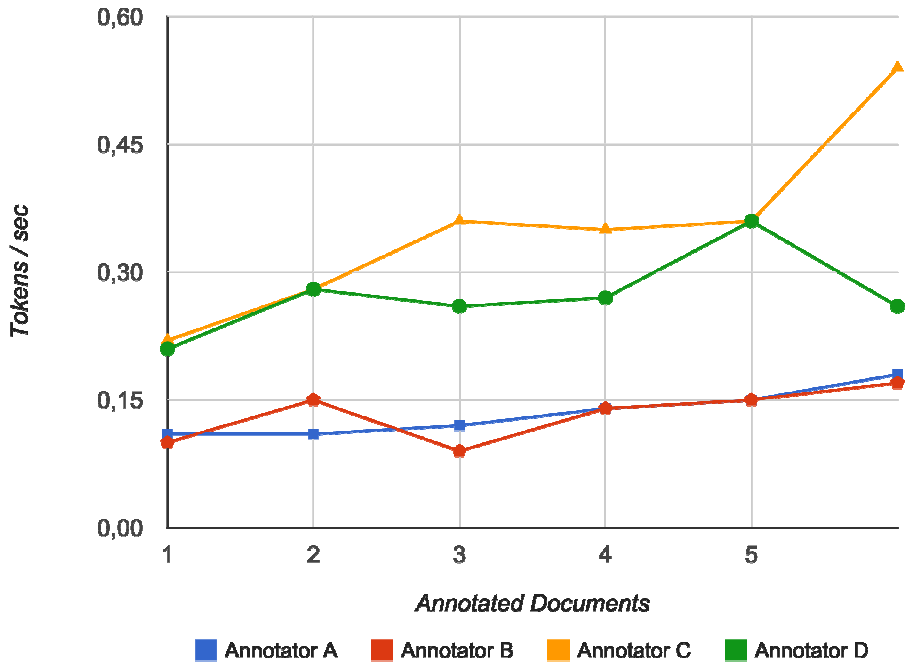

Figure 8: Annotation Speed for Part-of-Speech Tags

Figure 9 shows the accuracy of the POS tag suggestions. The accuracy of the suggestions increases from document to document. At the end of the experiment the accuracy reaches 90 percent. The increasing accuracy limits the task of annotating POS to reading and verifying suggestions. This limitation improves annotation speed further (c. f. increasing annotation speed of Annotator C and D in Figure 8).

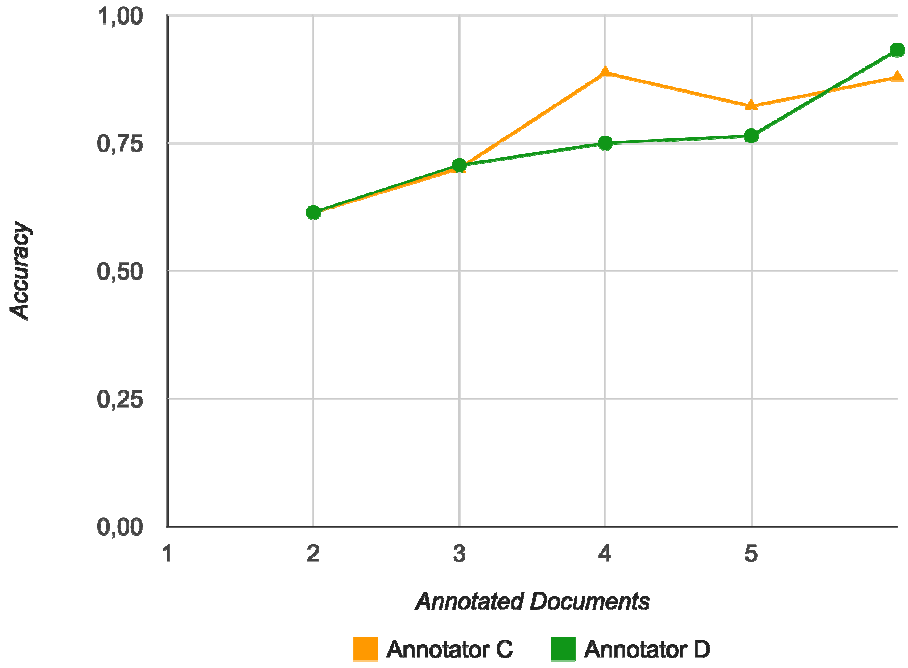

Figure 9: Accuracy Regarding Suggestions for Part-of-Speech Tags

The results of this experiment show that a custom GUI for annotating POS tags is worthwhile. Due to the custom GUI for the annotation layer the annotation speed could be improved by a factor of 2.5. The provided suggestions improve annotation speed further.

4.3. Named Entity Annotation Layer

Table 3 shows the average time spent to annotate NEs. On average the QPT reduces the annotation time by a factor of 2.4.

Table 3: Average Annotation Time for Annotating a Document with Named Entities

Webanno		**Quick Pad Tagger**	
Annotator A	Annotator B	Annotator C	Annotator D
7:05 min	5:41 min	2:42 min	2:42 min

Figure 10 shows the annotation speed for each document and annotator. On average the annotation speed is 0.8 tokens per second higher by using the QPT. The annotation speed for both tools stays more or less the same. When the third and fourth document are annotated, the annotation speed drops for both tools.

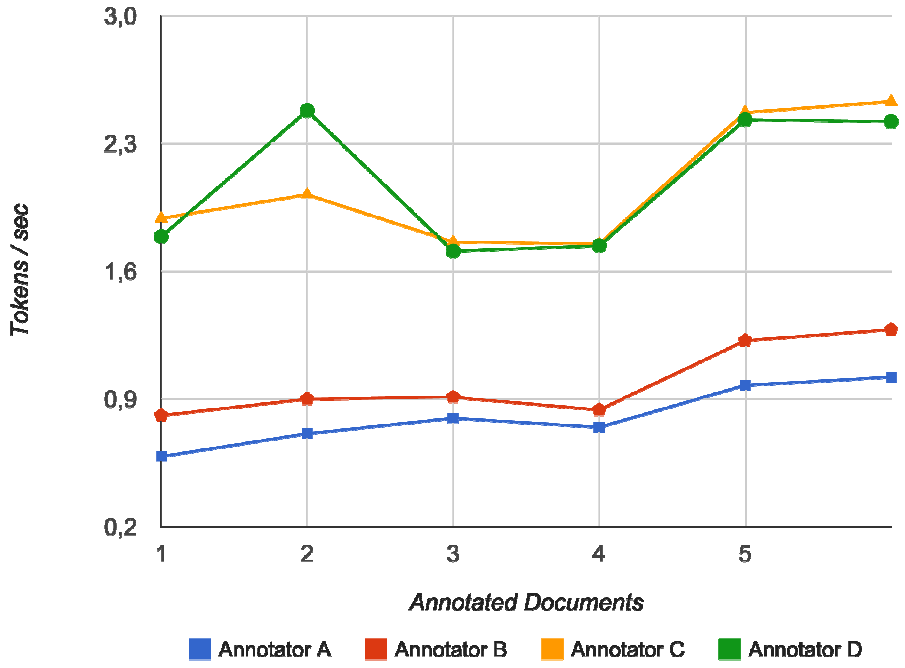

Figure 10: Annotation Speed for Named Entities

The drop of annotation speed for the third and fourth document has following reason: The annotators struggled independently to annotate some NEs and they asked for advice. The drop of annotation speed for Annotator C and D is more distinctive because on average it took not so much time to annotate the whole document (c. f. Table 3).

Additionally, in this experiment the annotation of the QPT was not influenced by the suggestions. Because of the small corpus size the F_1 score of the NE suggestions was zero. The suggestions made by the QPT were not helpful to annotators and in most cases the suggestions had to be removed.

In conclusion, the results of this experiment show a custom GUI for annotating NEs improves the annotation speed significantly (factor of 2.4). The annotation speed of the QPT outperforms the annotation speed of Webanno even with faulty suggestions. The F_1 score of NE suggestions was zero and did not boost the annotation speed further.

4. CONCLUSION AND FUTURE PROSPECTS

We presented the QPT and compared the QPT with Webanno regarding annotation speed. In our three experiments we showed that the QPT outperforms Webanno in terms of annotation speed because of two reasons:

1. The QPT provides for each annotation layer a specific GUI based on a keyboard input method. Webanno provides a generic GUI for all annotation layers based on a mouse input method. QPT's specific input methods are more efficient than Webanno's generic input method.
2. The version of Webanno used for testing does not provide suggestions to the annotator. The design of the QPT includes providing suggestions right from the beginning which enables another speed up regarding annotation speed.

Normally, users have to learn to use specialized GUIs which often takes a long time. Despite the specialized GUI, Annotator C and D were able to learn to use the QPT very quickly during our experiment. This verifies that the GUI of the QPT has a high efficiency and effectiveness [25]. Currently, the QPT is used to develop a large domain specific annotated corpus. In future work we will address Gamification elements [19] to improve the user experience furthermore. We expect that an improved user experience increases annotation speed further. Additionally, an improved user experience should keep the annotation quality high.

REFERENCES

[1] Roman Klinger, Corinna Kolarik, Juliane Fluck, Martin Hofmann-Apitius, and Christoph M. Friedrich. Detection of IUPAC and IUPAC-like Chemical Names. Bioinformatics, 24(13):i268–i276, 2008.

[2] Eugenie Giesbrecht and Stefan Evert. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. In Proceedings of the 5th Web as Corpus Workshop, WAC5, pages 27–35, 2009.

[3] Neil Barrett and Jens H. Weber-Jahnke. Building a biomedical tokenizer using the token lattice design pattern and the adapted Viterbi algorithm. BMC Bioinformatics, 12(S-3):S1, 2011.

[4] Haibin Liu, Tom Christiansen, William A. Baumgartner, and Karin Verspoor. BioLemmatizer: a lemmatization tool for morphological processing of biomedical text. Journal of biomedical semantics, 3, 2012.

[5] Jeffrey P Ferraro, Hal Daumé, Scott L DuVall, Wendy W Chapman, Henk Harkema, and Peter J Haug. Improving performance of natural language processing part-of-speech tagging on clinical narratives through domain adaptation. Journal of the American Medical Informatics Association, 2013.

[6] Tim Rocktäschel, Torsten Huber, Michael Weidlich, and Ulf Leser. WBI-NER: The impact of domain-specific features on the performance of identifying and classifying mentions of drugs. In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 356–363, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[7] Melanie Neunerdt, Bianka Trevisan, Michael Reyer, and Rudolf Mathar. Part-Of-Speech Tagging for Social Media Texts. In Iryna Gurevych, Chris Biemann, and Torsten Zesch, editors, Language Processing and Knowledge in the Web, volume 8105 of Lecture Notes in Computer Science, pages 139–150. Springer Berlin Heidelberg, 2013.

[8] Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, and Sebastian Pado. SALTO: A Versatile Multi-Level Annotation Tool. In Proceedings of LREC-2006, 2006.

[9] Philip V. Ogren. Knowtator: A Protégé plug-in for annotated corpus construction. In Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, NAACL-Demonstrations '06, pages 273–275. Association for Computational Linguistics, 2006.

[10] Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL 2013), pages 1–6, 2013.

[11] Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. Journal of the American Medical Informatics Association, 2013.

[12] Hrafn Loftsson, Jökull H. Yngvason, Sigrún Helgadóttiry, and Eiríkur Rögnvaldssonz. Developing a PoS-tagged corpus using existing tools. In Proceedings of "Creation and use of basic lexical resources for less-resourced languages", workshop at the 7th International Conference on Language Resources and Evaluation, 2010.

[13] Karën Fort and Benôıt Sagot. Influence of Pre-annotation on POS-tagged Corpus Development. In Proceedings of the Fourth Linguistic Annotation Workshop, LAW IV '10, pages 56–63. Association for Computational Linguistics, 2010.

[14] Andreas Mengel and Wolfgang Lezius. An XML-based representation format for syntactically annotated corpora. In In Proceedings of the International Conference on Language Resources and Evaluation, pages 121–126, 2000.

[15] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, and M. A. Musen. Protégé-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment. AMIA Annual Symposium Proceedings, pages 953+, 2003.

[16] Seid Muhie Yimam, Richard Eckart de Castilho, Iryna Gurevych, and Chris Biemann. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations, page (to appear), June 2014.

[17] David M. Lane, H. Albert Napier, S. Camille Peres, and Aniko Sandor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. Int. J. Hum. Comput. Interaction, 18(2):133–144, 2005.

[18] Richard C. Omanson, Craig S. Miller, Elizabeth Young, and David Schwantes. Comparison of Mouse and Keyboard Efficiency. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, pages 600–604, September 2010.

[19] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. Using Game-design Elements in Non-gaming Contexts. In CHI '11 Extended Abstracts on Human Factors in Computing Systems, CHI EA '11, pages 2425–2428, New York, NY, USA, 2011. ACM.

[20] Torsten Zesch, Christof Müller, and Iryna Gurevych. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In Proceedings of the Conference on Language Resources and Evaluation (LREC), electronic proceedings, May 2008.

[21] Welcome to Apache OpenNLP. https://opennlp.apache.org/.

[22] The Stanford Natural Language Processing Group . http://nlp.stanford.edu/software/corenlp.shtml.

[23] Slav Petrov, Dipanjan Das, and Ryan McDonald. A Universal Part-of-Speech Tagset. In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), may 2012.

[24] Ulrich Heid, Helmut Schmid, Kerstin Eckart, and Erhard Hinrichs. A Corpus Representation Format for Linguistic Web Services: The D-SPIN Text Corpus Format and its Relationship with ISO Standards. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), may 2010.

[25] Jeffrey Rubin and Dana Chisnell. Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests. Wiley Publishing, 2nd edition, 2008.