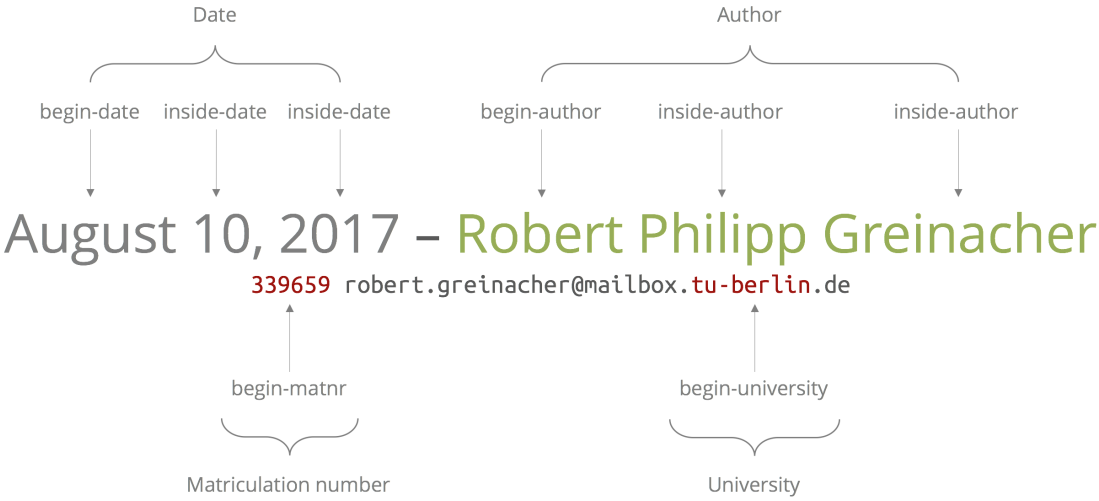
Technische Universität Berlin
Fakultät IV - Elektrotechnik und Informatik, sowie:
Fakultät V - Verkehrs- und Maschinensysteme

Bachelor's Thesis:

Efficiency Evaluation of an Assistance System for Text Annotation

Untersuchung der Effizienz eines Assistenzsystems für Textannotationen



Department of Psychology and Ergonomics:
Chair of Cognitive Psychology and Cognitive Ergonomics,
Institute of Software Engineering and Theoretical Computer Science:
Machine Learning

Supervisors:
Prof. Dr. Manfred Thüring and
Prof. Dr. Klaus-Robert Müller

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form bei keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Berlin, August 11, 2017

Ort, Datum Robert Philipp Greinacher

Abstract

Smart personal assistants like Siri or Alexa use machine learning based text analysis to understand commands and questions. Underlying technologies require a huge amount of training data and their production often involves human annotators doing repetitive and monotonous work. The purpose of the present study is to identify possibilities to support the text annotation task with automated assistance. Since these annotation assistance systems cannot be perfectly accurate the influence of reliability is analyzed, too. We developed a fully functional, autonomous assistance system, specialized on named entity recognition to prove the feasibility of such an assistance. It creates suggestions for annotators based on what it has learned from previous annotations. For the study we simulated such a system. Subjects ($N = 66$) accomplished annotation tasks with and without the system in a counterbalanced order. Three levels of reliability (10%, 50%, or 90% correct suggestions) of the system were benchmarked. Dimensions measured were correct annotations (hits), misses of annotations, and time per correct annotation. A 2×3 mixed design (assistance present / not present, within and the three levels of reliability, between) was used. An assistance system providing a reliability of 50% or 90% improves accuracy, average time per correct annotation, as well as reducing misses of the human annotations significantly. Supporting the task of text annotation by using an automated assistance will improve humans' performance if the suggestions of the system are mature.

Zusammenfassung

Persönliche Assistenzsysteme wie Siri oder Alexa nutzen Textanalysen, um Eingaben und Fragen zu verstehen. Die darunterlegende, auf maschinellem Lernen beruhende Technologie benötigt große Trainingsdatensätze, deren Erstellung oft mit monotonen und repetitiven Aufgaben für menschliche Annotatoren einhergeht. Ziel der vorliegenden Studie ist es, Möglichkeiten zu identifizieren, um die Textannotationen mit einem automatischen Assistenzsystem zu unterstützen. Da solche Systeme nicht unfehlbar sein können, wird untersucht, welchen Einfluss die Verlässlichkeit des Assistenzsystems hat. Wir entwickelten ein vollständig funktionierendes, auf Eigennamenerkennung spezialisiertes, autonomes Assistenzsystem, um die grundsätzliche Umsetzbarkeit einer solchen Assistenz zu beweisen. Das System lernt aus vorhergehenden Annotationen und generiert Vorannotationen in neuen Texten. Diese dienen als Vorschläge für die Annotatoren. Für die Studie wurde ein solches System simuliert. Die Probanden ($N = 66$) annotierten Texte abwechselnd mit und ohne Assistenzsystem. Dabei gab es drei Verlässlichkeitsstufen der Assistenz: Es lieferte 10%, 50%, oder 90% korrekte Vorschläge. Als abhängige Variablen wurden korrekte Annotationen, fehlende Annotationen und Zeit pro korrekter Annotation erhoben. Die Studie war in einem 2×3 gemischten Design (mit / ohne Assistenzsystem und die drei Stufen der Verlässlichkeit) aufgebaut. Wir zeigen, dass ein Assistenzsystem mit 50% oder 90% richtigen Vorschlägen bei den Probanden die Rate der richtigen Annotationen steigert, die Zeit pro korrekter Annotation senkt und die Fehleranzahl vermindert. Die menschliche Leistung bei Textannotationen kann durch ein automatisches Assistenzsystem verbessert werden — wenn es eine hohe Treffsicherheit in seinen Vorschlägen hat.

Acknowledgment

I would like to thank Franziska Horn and Dr. Nils Backhaus for their extraordinary support, patience and expert advice throughout the project. I would like to thank the team of the Chair of Cognitive Psychology and Cognitive Ergonomics of the Technische Universität Berlin for their steady willingness to discuss and advise me with any question I ever had. This project would have been impossible without the brilliant support of Hannes Korte, Dr. André Bergholz and Dr. Andreas Schäfer of the Implisense GmbH.

To Charlotte.

Contents

Acknowledgment V

Lists of Abbreviations VIII

List of Figures IX

List of Tables X

1. Introduction 1
- 1.1. Motivation . . . 1
- 1.2. Current State of Tools and Research . . . 3
- 1.3. This Approach . . . 6

2. An Annotation Framework with Assistance 9
- 2.1. Conception . . . 9
  - 2.1.1. Requirements of the Framework as a Whole . . . 9
  - 2.1.2. Active Learning at Heart . . . 11
  - 2.1.3. Data Flow Inside of DALPHI Active Learning Platform for Human Interaction . . . 12
- 2.2. Implementation . . . 14
  - 2.2.1. Web Service Architecture . . . 15
  - 2.2.2. Named Entity Recognition Using NLTK . . . 16
- 2.3. Simulation of the Assistance System for the Study . . . 21

3. Method 24
- 3.1. Sample . . . 24
- 3.2. Apparatus . . . 25
  - 3.2.1. Annotation Interface . . . 25
  - 3.2.2. Assistance System . . . 26
  - 3.2.3. Questionnaires . . . 26
- 3.3. Procedure . . . 27
- 3.4. Data Analysis . . . 28
- 3.5. Experimental Design and Variables . . . 28
- 3.6. Hypotheses . . . 30

4. Results 32
- 4.1. Impact of the Assistance System on the Three Performance Parameters . . . 32
  - 4.1.1. Impact of the Assistance on Correctness . . . 32
  - 4.1.2. Impact of the Assistance on Tempo . . . 33
  - 4.1.3. Impact of the Assistance on Misses . . . 35
- 4.2. Differences of the Three Levels of Assistance . . . 36
  - 4.2.1. Correctness . . . 36
  - 4.2.2. Tempo . . . 37

4.2.3. Misses . . . 38
4.3. Questionnaires . . . 38
4.3.1. Perception of Workload . . . 39
4.3.2. Perception of Monotony . . . 40
4.3.3. Perception of Reliability . . . 41
4.3.4. Perception of Correctness . . . 43

5. Discussion **45**
5.1. Hypotheses . . . 45
5.1.1. Questionnaire Evaluation: Perceived Workload . . . 49
5.1.2. Questionnaire Evaluation: Perceived Monotony . . . 50
5.1.3. Questionnaire Evaluation: Perceived Reliability of the Assistance . 50
5.1.4. Questionnaire Evaluation: Perceived Correctness of the Assistance 51
5.2. Method Discussion . . . 51
5.3. Practical Implications . . . 52
5.4. Further Research Questions . . . 53
5.5. Conclusion . . . 53

6. Appendix **59**

A. Analysis Scripts and the Data Set **59**

B. The Development of the DALPHI Active Learning Platform for Human Interaction (DALPHI) framework **59**

C. The Study Texts **60**

D. The Questionnaires **62**

E. Study Material **64**
E.1. Declaration of Consent . . . 64
E.2. Annotation Guidelines . . . 65
E.3. User interface Functions . . . 67

Lists of Abbreviations

AL Active Learning

ANOVA Analysis Of Variance

API Application Programming Interface

BFGS Broyden–Fletcher–Goldfarb–Shanno

CRF Conditional Random Field

CSS Cascading Style Sheet

DALPHI DALPHI Active Learning Platform for Human Interaction

DB database

DUALIST Utility for Active Learning with Instances and Semantic Terms

GUI Graphical User Interface

HTML Hypertext Markup Language

JS Java Script

JSON Java Script Object Notation

MEC Maximum Entropy Classifier

ML Machine Learning

NER Named Entity Recognition

NLP Natural Language Processing

NLTK Natural Language Toolkit

POS Part Of Speech

UI User Interface

List of Figures

1. Smart assistants use Machine Learning (ML) to process spoken requests. . 1
2. Exemplary analysis of a simple sentence. 2
3. Example of a German text with and without annotated entities. Green annotations identify person names, blue annotations identify company and organization names. 3
4. Examples of single-user annotation tools. 4
5. Two popular web-based and collaborative text annotations tools. 4
6. The annotation workflow: The assistance system works on the plain text before it will be displayed to a human annotator. Its impact should improve the work of the human. 8
7. Visualization of the DALPHI Active Learning Platform for Human Interaction work cycle. 9
8. The processing and data flow cycle of the DALPHI framework. Green documents are annotated, black documents are not. 12
9. A word pair example with an ambiguous meaning. 17
10. A named entity tagged example sentence using the beginning / inside / outside notation. 18
11. The DALPHI User Interface (UI) for text annotations. 25
12. Four blocks of the study, the assistance's presence was toggled. 26
13. The phases of the study for all groups. 29
14. All three levels of assistance regarding the performance dimension correctness. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. 33
15. All three levels of assistance regarding the performance dimension correctness. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. 34
16. All three levels of assistance regarding the performance dimension misses. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. 35
17. All three levels of assistance regarding the perceived workload of the subjects. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. 39
18. All three levels of assistance regarding the perceived monotony of the subjects. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. 41
19. All three levels of assistance regarding the perceived reliability of the assistance system. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. . . 42
20. All three levels of assistance regarding the perceived correctness of the assistance system. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions. . . 43

21. Intermediate questionnaire with four items, presented after blocks with assistance. 62
22. Demographic questionnaire that was asked after the fourth block of trials. 63

List of Tables

1. Scores of the Maximum Entropy Classifier model evaluation, compared with results from the GermEval 2014 (GE) competition [24]. 21
2. Possible annotation errors. Green labels are person names, blue labels are company names. 22
3. Evaluation of the simulated assistances and our Maximum Entropy Classifier model. 23
4. Summary of the hypotheses 32
5. Results of the Analysis Of Variance (ANOVA), analyzing the assistance system's influence on correctness. 36
6. Results of the ANOVA, analyzing the assistance system's influence on tempo. 37
7. Results of the ANOVA, analyzing the assistance system's influence on the rate of missed annotations. 38
8. Results of the ANOVA, analyzing the perception of workload (via questionnaires). 40
9. Results of the ANOVA, analyzing the perception of reliability (via questionnaires). 42
10. Results of the ANOVA, analyzing the perception of correctness (via questionnaires). 44
11. List of hypotheses, highlighting significant results. 48

1. Introduction

This section explains the basic terms we are going to use in this work. We first motivate the topic of text annotation and therefore will explain related Machine Learning (ML) issues and applications. Next, current tools and research are explained. The section closes with a detailed description of how we want to improve the current situation of training data generation and what we are going to present in the following chapters.

1.1. Motivation

The three main terms of the title of this work are Text annotation, assistance and evaluation. To motivate and explain the relation of these terms, we are going to have a look at their meaning.

(a) Apple Siri (b) Google Home (c) Facebook M (d) Amazon Alexa (e) Microsoft Cortana

Figure 1: Smart assistants use ML to process spoken requests.

Using ML allows data scientists to extract information from huge amounts of data. Named Entity Recognition (NER), for example, is such an extraction technique: The goal is to identify names of persons, places, groups (like organizations or companies) and other entities in texts. Applications to such technology are for example smart assistances like Siri on the iPhone or Alexa as a standalone device in living rooms. Five of the most popular examples of such assistances are listed in Figure 1. They all have in common to first translate speech to text and then process this text to *understand* what it is about. Smart assistances aren't the only field of application, though: Chatbots and spam filters use NER as well, to name just a few end user products. This kind of technology is also useful when it comes to the analysis of large amount of data, exceeding the amount humans could process in a reasonable amount of time. Another real world example: Leaks like the Snowden documents of the Panama Papers need to be processed automatically to get an idea of where to start an investigation, since such large scales of text are simply too big to be assessed by an individual in a reasonable amount of time.

Tools and techniques that help creating machines capabale of processing natural language text are summarized as Natural Language Processing (NLP). The task of automatically recognizing named entities is called NER. This lays an important foundation to make machines *understand* human sentences. One way to teach a machine is by presenting both the data that needs to be processed and the result of a processing to it. The *how*

can be inferred. An ML algorithm tries to generalize the information from the training (the learning phase) and creates a *model*, which will later be used to predict an outcome towards data it has not seen before. In this case it means showing the algorithm texts with correctly annotated entities. This presentation of showing data and labels at once is called *supervised learning*.[1]

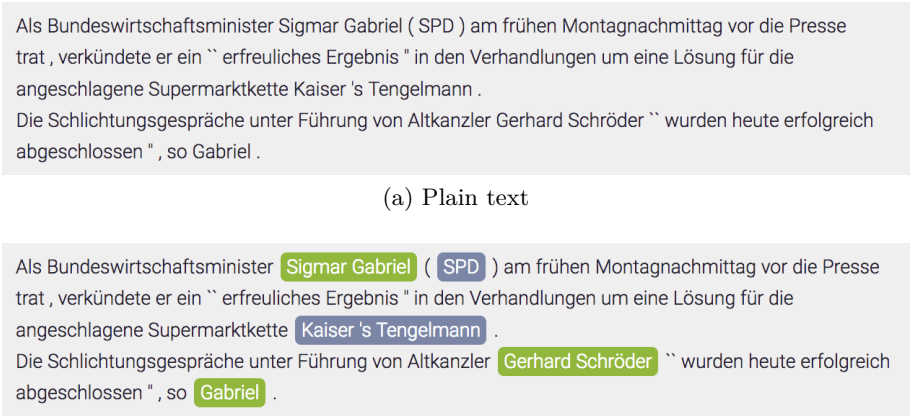
It is hard to create a model that reflects complex concepts like understanding a natural language. Figure 2 shows a few of the fundamental steps needed to access the meaning of a rather simple sentence. A common concept to handle hard problems like this is to divide it into several simpler subtasks. The chain of these subtasks is called *pipeline*. The final outcome of such a pipeline depends much on the quality of each of the single steps. This is why good ML models depend on good training data – everything they "know" has been inferred from the training data. If this data doesn't depict a certain aspect of the desired behavior a resulting model has no chance to make correct predictions. Training data is key to the model's performance. In the case of ML for NER training data are annotated texts.

Figure 2: Exemplary analysis of a simple sentence.

Text Annotation Annotated texts are one or more words or phrases labelled and often highlighted, depending on the UI. These annotations have special meanings, abstractly describing the membership of a group. For example to identify names of companies, names of places, addresses or phone numbers – to name just a few. In this work we are interested in the annotation of named entities, like in the example given in Figure 3:

ML models working on natural language are most of the time specifically trained for one language, often even for a certain text domain. They work by deriving patterns from their training data. Hence they can not be (usefully) used for an other language. If text domains heavily drift apart from each other in terms of grammar or word use, a model trained on one text domain might not work well on another one, even if both are in the same language. For example, an NER model trained on scientific papers used on Twitter

[1] In contrast to *unsupervised learning*. This describes a group of algorithms which try to learn something (like patterns for classification for example) out of unlabeled data.



(a) Plain text

(b) Annotated text

Figure 3: Example of a German text with and without annotated entities. Green annotations identify person names, blue annotations identify company and organization names.

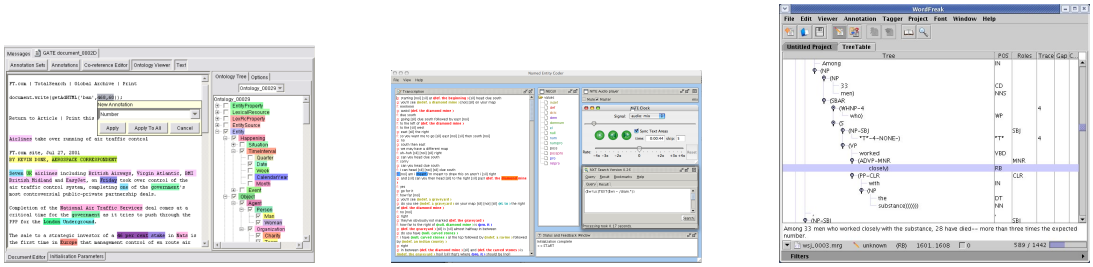
tweets will most likely not perform as well as one trained on the appropriate data set – even though the same algorithm is used.

To generate different models for different text domains, data scientists need great amounts of different training data sets. Each of them need to be carefully crafted, as correct and complete as possible, allowing the derivation of as much information as possible. This is mostly a manual and thus monotonous and time consuming task – and therefore it is expensive.

1.2. Current State of Tools and Research

There are various annotation tools designed to facilitate the generation of annotated text data sets. They are dividable into single-user and multi-user tools, i.e. for users who work alone or on only one computer and for users who want to distribute the annotation task to more than one annotator. First generations of Graphical User Interface (GUI) annotation tools were single-user based like for example *GATE Developer* [1] (started in 1995), the *NITE XML Toolkit* [2] (2002) or *WordFreak* [3] (2003) (see Figure 4). They all run on computers locally and are designed as individual systems, not needing a server in the background to operate. Many of them are written in Java, which makes them more platform compatible than natively complied applications. They still require to be installed on each individual computer, though.

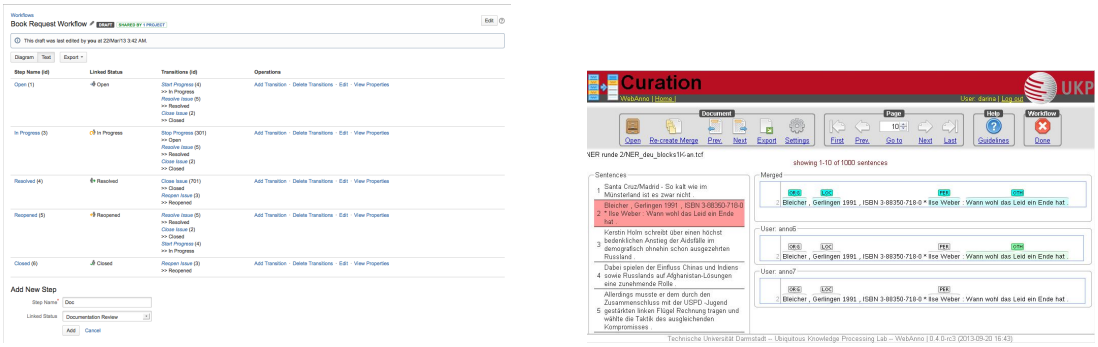
In most cases modern tools belong to the multi-user class since modern web technologies have developed rapidly towards superseding native applications. Without a need for local software installation, browser based applications offer a fast access to every annotator using a browser. This reduces the requirements to participate and makes col-



(a) GATE Developer (b) NITE XML Toolkit (c) WordFreak

Figure 4: Examples of single-user annotation tools.

laboration a lot easier. Collaboration is an often demanded feature to be able distribute up the workload to not only one but many annotators. Biemann et al. [4] provide a comprehensive overview of web-based, collaborative annotation tools. Not all but most of them focus on text annotation (highlighting spans and relations between spans) and thus would be in line for training data generation for the example task of this work, NER. Two frequently used and often cited tools are *GATE Teamware* [5] and *WebAnno* [6] (see Figure 5). Another often cited web-based annotation application is Brat [7], which is based on Stav [8], an annotation visualizer from the same developers.



(a) GATE Teamware (b) WebAnno

Figure 5: Two popular web-based and collaborative text annotations tools.

There are various tools serving different needs like running isolated without network connectivity or as a network application with collaboration and different user roles for administration, curation and annotation. Even tools focusing on flexibility and customization with add-ons are available [4]. Some of the mentioned examples do support the the human annotator with suggestions that should help annotators doing their task: They prepare the data before the annotation and pre-label (or "pre-annotate") the text to assist the human. For example GATE Teamware and WebAnno do support this feature, partly built-in and partly as a service component. It seems plausible that pre-annotations could provide a mental guidance by presenting examples for the human annotators, reduce mouse movement and thus time because not all the annotations that

have to be made have to be clicked.

Day et al. [9] showed as early as in 1997 that annotator productivity can be enhanced by a support system generating pre-annotations. They used a mixed-initiative setup allowing their annotators to define a rule set for pre-annotation on their own and tried to derive annotation rules from a small text set that the annotators processed before they started the annotation of the complete corpus. The derived rules were used to pre-annotate the complete corpus. The group showed an increased annotation productivity when using such pre-processed text sets, measured in processed words per minute and annotated tags per minute.

The pre-annotation features of systems like the ones mentioned above are most of the time ML based and try to learn from already made annotations [4]. In contrast to our approach these ML modules use pre-trained models to predict labels. As a result, they are rather inflexible to perform on very different text domains. Moreover, they only provide pre-annotations instead of combining the output of their predictions with an Active Learning (AL) approach to annotate especially the sentences and paragraphs in which the model is uncertain about their annotation (see Section 1.3 for further explanation). One example of an annotation tool supporting this idea is Utility for Active Learning with Instances and Semantic Terms (DUALIST) by Burr Settles [10]. By using AL and a specifically created semi-supervised learning algorithm, annotators were able to create training data sets that achieved up to 90% of the then state-of-the-art performances in only a few minutes. Settles demonstrated the usefulness of an AL approach to create ML models to complement the strengths of both machine and annotator [10].

What is missing A combination of Day's [9] and Settle's [10] work is desirable; a system that uses pre-annotations to assist the human doing the task and that utilizes the advantages of AL and an iterative annotation cycle – not only for a certain use case but for any kind of data. A validation of the impact of pre-annotations to justify developmental efforts and to be able to describe the impact on the human instead of superficial outcomes (like processed words per minute) is missing as well. As a consequence, the available work encouraged us to combine both methodologies to create a new annotation framework and to facilitate a comprehensive study: To understand the impact of pre-annotations within this new annotation environment and to go more into detail about the affected performance dimensions of human annotators.

1.3. This Approach

To improve the task of how training data are generated the whole process of how such data are crafted has to be analyzed. Besides the impact of the UI that is used to annotate data, the annotation process of currently available tools itself is what we challenge. We developed a new general purpose annotation framework: It is customizable to fit any kind of data (format) and desired UI and provides an annotation work cycle to support AL and (constantly improving) pre-annotations from a ML unit. These suggestions should assist the annotators and thus increase efficiency. We evaluated this benefit in detail in Section 3 et seqq.

Research Question of This Work We aimed to improve the task of training data generation by asking ourselves how this time consuming process can be made more comfortable and faster. We want to assist the annotators doing their task. We want to increase the efficiency of the task of annotating texts. Besides many UI improvements we focused on the idea of shifting the task from annotating to supervising. This way the research question we developed was: Can a system compute annotations that help to improve the task of manual data labeling? If we utilize an AL approach to support annotators, will it result in more correct, faster and more complete annotations? This leads to the question of what efficiency means in this context and how it is operationalized.

Efficiency and Ways of Increasing it To define the word efficiency first and to delimit it from effectiveness:

> It is fundamentally the confusion between effectiveness and efficiency that stands between doing the right things and doing things right.
>
> *Peter F. Drucker* [11]

Or in other words: Effectiveness is a measurement of reaching a goal altogether, whereas efficiency describes wasting no resources while doing a certain task or creating as much of a desired outcome as possible.
Increasing efficiency therefore means to gain more outcome for the same effort, or for the mission of improving training data generation: Getting *more* text annotated in the same amount of time (reducing the time spent per annotation) or getting more *correct* annotations done in the same time – and in the best case both at once. As seen in Section 1.2, the available tools often lack a clean, simple and straight forward UI: They are cluttered with control elements and because they show the whole document at once (rather than only singular extracts to work on at a time) it is hard to focus on what is important (which would be only one sentence and its corresponding labels).

Reducing the number of functions and control elements in the UI to a minimum (nothing but the necessary), not overloading the annotators with information but rather presenting one text extract at a time is in line with usability guidelines to improve the usability of such an interface [12].[2]

Another way to improve annotation efficiency can be done by computing a metric of importance ranking each part of the whole data. All the documents that need to be annotated can then be ordered by using this metric in a *most-impact-first* manner. Creating training data sets that contain only data that would "help" the model to improve a lot would make it unnecessary for annotators to process passages that would not improve the model significantly. This intentional skipping of non-effective work would save time and thus improve efficiency. Many approaches to this metric are possible: A model can be trained and a confidence score for each label prediction can be calculated. A very low confidence would indicate a necessary human examination. However, this work focuses on the third element we identified as important for a new annotation environment: the annotation assistance system.

The Assistance System To answer the research question this work focuses on the assistance system. In particular: on how it helps to improve the efficiency of the annotation task. As already mentioned, the system should make suggestions of annotations that only have to be approved by a human annotator. Our main hypothesis was that overseeing suggestions and correcting them as appropriate will lower the cognitive demands of the annotators and therefore accelerate and improve the annotation task with respect to correctness.

If we can achieve this, we would be able to create more training data sets, since we generally lower the cost of producing them. This would allow us to create less general but more domain specific training data, which would impact the performance of current ML applications: For example the performance of NER systems on German texts is worse than the one on English texts, on one part because for English texts there is far more training data available [13]. The performance could be improved if we could train several specialized NER models for different text or topic domains. This is currently done infrequently due to a lack of diverse but large enough training data sets. If our hypothesis about our annotation framework holds, we would be able to create training data more rapidly and therefore more specialized for their applications.

The general idea of the assistance, from a procedural point of view, is to predict new annotations (as suggestions) to lighten the workload of the annotators. See Figure 6 for a schematic flow. The human work as well as the assistance system itself is a source of error. We nonetheless assume that the assistance system has a positive influence on the quality of the annotated texts and the time spent by the human annotator.

[2] We aimed to provide a full stack efficiency improving tool chain with our annotation framework, at least for training data generation for the NER task. Therefore we developed a UI with these guidelines in mind. It remains to the interested reader to evaluate this interface to see if our attempt of improving efficiency with our UI was successful.

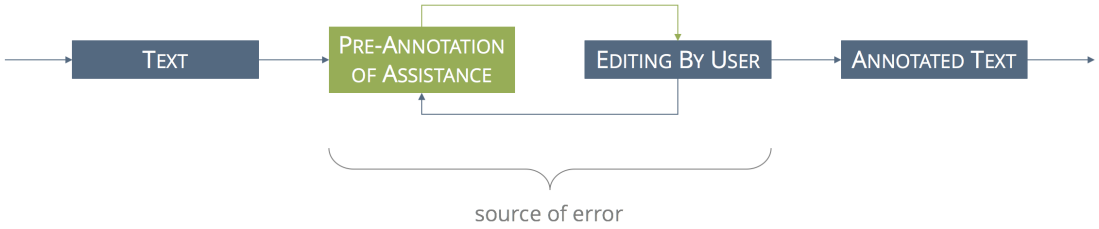


Figure 6: The annotation workflow: The assistance system works on the plain text before it will be displayed to a human annotator. Its impact should improve the work of the human.

To reach this goal we want to split the entire process into several smaller parts: After each iteration the assistance can learn more and make better predictions over time. Such an assistance system will make mistakes. Its prediction will get better over time, but it will never be perfect (if it were perfect, there would be no need for training data to create such a perfect system). We assume that such an assistance, even if it makes mistakes, will nevertheless assist the human (see hypothesis in Section 3.6).

The rest of this thesis is structured as follows: We describe the conception, the implementation and the mathematical background of our implementation of an assistance system providing named entity pre-annotations. This implementation serves as an example of the assistance as described. We continue with a detailed explanation of how we conducted the study to evaluate the impact of such an assistance on the human annotation performance. Subsequently we present the results of this evaluation and a detailed discussion of these results.

2. An Annotation Framework with Assistance

We constructed an annotation environment that supports the manual task of annotating data using ML to assist the annotator. We explain its mechanics at the example of creating training data for NER (Section 2.1.2). However, the annotation environment is modular and thus capable of creating training data for other applications as well. We discuss the architectural key points of the annotation framework (Section 2.2.1) and, as a proof of concept, we demonstrate an ML based assistance system using an Maximum Entropy Classifier (MEC) to identify named entities (Section 2.2.2). Finally we explain the mathematical background of the assistance (Section 2.2.2) we implemented.

Our annotation framework DALPHI was designed to assist the human doing the annotations with pre-annotations, AL support, and offers an iterative work cycle. A software framework that helps to improve the task of training data generation has several different requirements to meet. It is hard to understand the design decisions that have impacted the assistance system component without considering the design decisions for the whole annotation framework. Therefore the following sections will focus on the assistance system, but also provides insights into the design process of the DALPHI framework as a whole.

2.1. Conception

Understanding the needs of the users of this tool is key to make it helpful. Target users of this software are data scientists (the annotation process itself can be sourced out to be done by non-professionals as well). We expect the user group to know some programming as we require them to convert their data so that it matches our interfaces. This allows us to construct the framework in a more flexible manner as we can expect our users to adjust their own software to the framework's interfaces.

2.1.1. Requirements of the Framework as a Whole

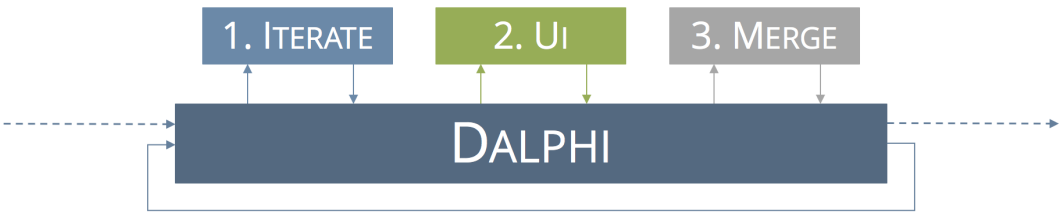


Figure 7: Visualization of the DALPHI Active Learning Platform for Human Interaction work cycle.

Most of all, the framework should provide a unified workflow cycle (illustrated in Figure 7): Uploading and organizing data, annotating data, processing data and again starting a new iteration of annotating data are the core steps. It is not about annotating everything at once, it is about dividing the task into several iterations. Annotators will not be overwhelmed by the amount of data to annotate. This allows the system to be retrained on the new data and to then make predictions between each iteration. One iteration refers to the time an annotator spends on annotating data. This period can cover the annotation of many, but at least one complete document.

The suggestions of the assistance are not created live in the very moment in which an annotator is working with the data, but after s/he is done with the current iteration. The annotator can decide to stop an ongoing iteration after each processed document. When an iteration finishes, the systems will take all the annotated data from the just finished iteration (and all the previous iterations), create a model from these annotations and make predictions about the remaining, not already annotated data. These documents will be reviewed by the human annotator in the next iteration.

In the following iteration, again a fraction of the whole data will be annotated. But this time, the human annotator is correcting wrong predictions and completing misses of the system. After this second iteration, the assistance component will again learn from the changes made and improve itself; this time it can create an even better model from the data because it has more and refined information in the training data. This way it will make more accurate predictions and will find annotations it has missed in the previous iteration – annotations that haven't been reviewed by human annotators yet.

To provide this learning mechanism not only for NER related data, the crucial elements (the UI, the assistance component) have to be problem agnostic. The whole framework should be suitable to create training data for any kind of problem. This implies that the mentioned components are not shipped with the framework; they have to be written by the users of the framework.[3] Thus the Application Programming Interface (API)s need to be relatively easy to adapt in order to lower the adaption costs for users. Other important design decisions are:

1. The framework has to be modular to easily change crucial components, like the UI or the assistance system. This is needed for rapid testing during the development of these components and in order to be flexible for different tasks.
2. The assistance should be invisible to the user (a background task). It should learn without any preparation or intervention by the user.
3. The framework's UI and APIs should be as simple as possible.
4. To satisfy the demand of flexibility (create training data not only for NER but for all supervised ML tasks), this UI should be easily exchange- or replaceable.

[3] We provide an OpenSource collection of examples for different use cases. They can be modified and adapted – and provide an easy access to the framework.

5. The framework should be open source from the very beginning. This hopefully helps growing the user base and getting contributions from other data scientists to grow the number of available ML and UI components and increase the feature set of the existing ones. We hope to receive collections of training data sets as well, all using the same format to provide as many tools and resources for new users as possible.

We created the DALPHI framework having the requirements above in mind. The framework itself and its components are free software and can be found on GitHub.[4]

2.1.2. Active Learning at Heart

The key requirement for the assistance is to learn and to improve iteratively. It uses the provided ML algorithm frequently to train a better model with each step.[5] That said, the assistance is nothing but the multiple (iterative) application of the same ML technique. In the case of training data generation for NER on which we focus within this work, the underlying algorithm for the assistance system is one of the standard NER classifiers (see Section 2.2.2 for an explanation of the classifier we used in detail).[6] A NER classifier labels every word of a text as a named entity (if any) [14]. Usual labels are names of people, organizations / company names or locations [15].

The way the ML algorithms are used is similar to a classical Active Learning (AL) approach: The machine predicts new labels and a human supervisor controls the process by correcting, adding or accepting the predictions of the model [16], [17]. Furthermore the human will only be asked if the algorithm's confidence for one or more of the pre-annotations that are presented at once are below a certain threshold. This way the human "helps out" in difficult situations whereas the algorithm will label the data in easy cases on its own. Since this workflow can be applied to many different kinds of ML training data generation, this assistance / AL combination is where the acronym DALPHI Active Learning Platform for Human Interaction comes from.

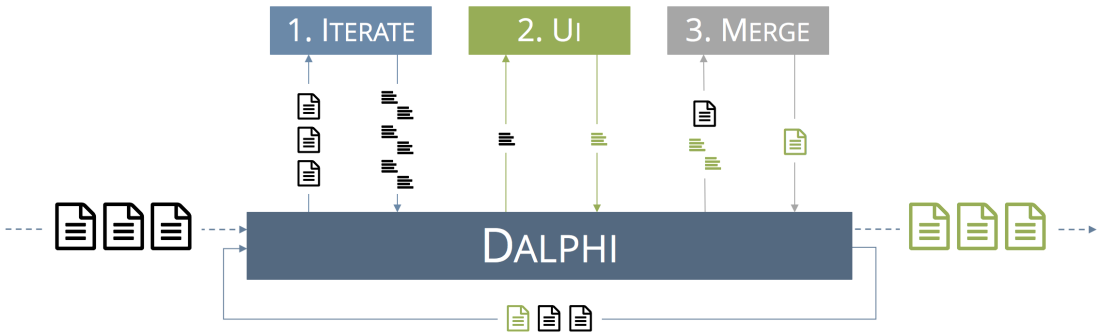


Figure 8: The processing and data flow cycle of the DALPHI framework. Green documents are annotated, black documents are not.

2.1.3. Data Flow Inside of DALPHI Active Learning Platform for Human Interaction

Figure 8 describes the four parts the DALPHI framework consists of: The framework itself with its APIs, an Iteration Service (1.), an annotation UI (2.) and a merge component (3.) – all of which we will discuss in detail. The numbered components (1. - 3.) have to be provided by the framework's users. This is because only they themselves will know the structure of their data and how they have to be processed. Most of all, the DALPHI framework provides a workflow and unifying API-interfaces.

The first task of using the framework will be uploading all the data that should be annotated via the web interface of the framework itself. These uploaded data are called *raw data* (pure, raw data without any annotations). The following steps describe the first iteration (in the NER scenario):

1. The Iterate Service with the Assistance Component The iteration service will convert and split the whole corpus (the set of all the texts that should be annotated) it received into many *annotation documents*. This is because it can be assumed that the amount of all of the *raw data* is a lot more than one annotator could handle at once. So the whole corpus is being divided into many small passages a human annotator is easily capable of annotating. What the exact rule of division is depends on the data. As a rule of thumb, whole paragraphs will work: On the one hand they contain most information

[4]DALPHI on GitHub: `https://github.com/Dalphi` (last date of retrieval: August 11, 2017) Further details on the development and can be found in the Appendix B.

[5]The time to train a model depends heavily on the task and the algorithm. Some need only seconds, other several hours or even days for one training phase. In such cases project owners are likely to start a new iteration only if a useful amount of new annotations were gathered. Therefore DALPHI provides different user roles: An annotator account is not able to start a new iteration, only an admin account is. This way only project responsible members decide when to start new iterations.

[6]This classifier is exchangeable and could easily be replaced by another NER classifier or a completely different algorithm for a different use case.

needed to understand what a passage is about (they provide some context), on the other hand they are usually not too large.

Together with this splitting process, the assistance component will be applied to the yet unlabeled *annotation documents* for the first time – if it has a built in bootstrapping fallback: The idea here is to provide a pre-trained model that can predict labels even if the data set is fully unlabeled. Such a general purpose model (not specialized on a certain text domain) will not perform as good as a specialized model but it would catalyze the process in the beginning. After the data set has been labeled to an extent that a model based on the data set would perform better than the general purpose model, the iteration component would stop using the bootstrap method and continue with the self trained model instead. If the implementation of the assistance does not have such a bootstrapping component, it will simply return each *annotation document* without any changes.

2. The User Interface A UI, compatible with the created *annotation documents*, has to be provided to render the documents. This interface consists of two to three files: An Hypertext Markup Language (HTML) file is the basis of the interface and defines a hierarchical structure of everything that is visible within the interface. A Java Script (JS) file will provide client side software (which runs in the browser of the annotators) to handle user interactions like mouse clicks. It furthermore contains an encoding function, which we'll discuss in detail below. Optionally (but most certainly) a Cascading Style Sheet (CSS) file can be provided. This file contains styling instructions defining how elements of the HTML file should look like (and to a certain extend how they should behave as well, e.g. hover effects or animations). Using these files, the interface renders the text of each document and integrates control elements to manipulate or generate annotations. It acts like a template, filling itself with data from the framework.

Therefore, the framework's client side first loads all the interface files and then requests the next *annotation document* from the framework's database (DB).[7] It then renders the UI with the data of the current *annotation document*. After the annotation of one document is complete, the UI's annotation encoder is called to transform all the changes that have visually been made with the UI back into the *annotation document* format. The particular encoding is part of the UI component because the author of the interface code knows best how to transform the data back into the exchange format. The only requirement the DALPHI framework makes about this data format is to use Java Script Object Notation (JSON). JSON is a commonly used exchange format in web applications, mostly for structured text data (but it's even possible to use it for binary files like images if such files get pre-encoded using Base64 for example).[8]

[7] The *next annotation document* is always one that has not been processed yet and might follow an order like the importance of a document. The DALPHI framework provides a corresponding API endpoint.

[8] Although it should generally be avoided to use binary files in JSON code because of the overhead of the special encoding and its consequential need of additional computing resources.

The annotation UI processes only one *annotation document* at a time. After one document is annotated and saved, the encoded text annotation combination will be handed back to the framework's DB; the next document will be loaded and annotated. This process is repeated until the user decides that the first annotation iteration is complete.

3. The Merge Service If the iteration is finished – the annotator decided that the annotation process is completed for now – all the annotated *annotation documents*, together with their belonging *raw data* documents, are sent to a *Merge Service*. This is to merge the annotations (that are stored in the *annotation documents*) into the corpus (in which users are actually interested in). Up until this point the corpus is not annotated. After the merging task is completed, all the current *annotation documents* are deleted and the corpus is updated.

To summarize this process: One iteration entails splitting the corpus in processable, small blocks, annotating them and finally merging them again with the original data set.

When the next iteration starts, the now partly annotated corpus is sent to the assistance system in order to be split and pre-annotated again. This component will now train a model with the annotations that the corpus now contains, make predictions about not annotated entities in the text and generates a new pool of *annotation documents*. This process is repeated until the whole text is annotated or a certain model performance can be achieved with the training data.[9]

2.2. Implementation

From the very beginning on, the DALPHI framework was planned as an open source software. The appendix B includes a further explanation of the development process, including a description of who was involved and who funded the development.

The software is a WebApp, based on the Ruby on Rails framework.[10] This offers a modern, flexibel and convenient development environment and the fact that the resulting product is a WebApp satisfies the requirement of making its use platform independent and inherently available to everyone at once having access to the internet and a web browser [18]. Thus it belongs to the multi-user class of annotation tools.

Since the main focus of this work is on the impact of the assistance to human annotators (Section 4), its software core (the MEC) is treated as a black box, using the Natural Language Toolkit (NLTK) as an off-the-shelf NLP library.[11]

[9] To measure how well the model currently is, a further component can be integrated into the framework. This would take the corpus as it is and train one or more models (with different algorithms) which will be cross-validated to return a performance index like an F-score or a precision / recall ratio.
[10] Ruby on Rails: `http://rubyonrails.org/` (last date of retrieval: August 11, 2017)
[11] NLTK: `http://www.nltk.org/` (last date of retrieval: August 11, 2017)

2.2.1. Web Service Architecture

DALPHI should act as a general purpose annotation framework, flexible enough to accommodate many different applications. As such, it is problem agnostic and the core components (UI, ML component / assistance and a merge service as described in 2.1.3) are not supplied.[12] The need to write ones own (problem specific) code creates the necessity to make these components as easy to write as possible. We created and open sourced examples of all components and they are easy to adapt to be used in ones own scenarios.[13] Instead of including them into a large monolithic code base, these components are accessed using a simple RESTful API [19]. This allows users to rapidly prototype their code and does not restrict them to any specific programming language or toolchain; they don't have to leave the environment they are used to. The only requirement is a rather simple HTTP interface and the convention that all transmitted data have to be JSON [20] encoded.

The assistance system is a stand alone web service. It is called by the DALPHI framework after an annotation iteration has finished and the newly annotated passages are already merged back to into the corpus. This (JSON encoded) corpus is now sent via an asynchronous HTTP-POST request, together with a unique callback ID, to the web service.[14] The web service immediately returns a HTTP success status code after the transmission has ended successfully. It now extracts all the sentences containing at least one annotation and trains an NER model using NLTK (see Section 2.2.2). The resulting model is subsequently used to label all the parts of the corpus that have not been annotated by a human. Now the resulting model has all the human made annotations and (hopefully) some more, created by the NER model. This annotated corpus is now split up into several *annotation documents* that will be evaluated by (a) human annotator(s) in the next iteration. The web service may order these documents as well: Such an order could define for which documents the algorithm wants human feedback (the AL approach, see Section 2.1.2). Finally these documents are returned to the DALPHI framework, using an HTTP-POST response request to the callback url the framework provided with its initial call.

[12] At least not all of them: The used components for labeling training data for NER tasks are fully open source and may provide a good starting point to create own services for the framework (interface: `https://github.com/Dalphi/interface-ner_complete` (last date of retrieval: August 11, 2017) and ML and merge service component: `https://github.com/Dalphi/service-ner_iterate` (last date of retrieval: August 11, 2017)).

[13] We provide three examples of an annotation UI, for a very basic full paragraph classification with only a few lines of code to a more complex NER annotation interface with different user interactions like mouse movements and keyboard shortcuts. Our example iteration service provides the complete classifier for examplatory purposes, but the necessary DALPHI API calls are written in only one simple file.

[14] A unique callback ID is necessary because of the asynchronous request: The data size of the corpus can be that large, that the time of the initial transmission, the processing and the transmission of the answer back to the framework would cause HTTP timeouts. To avoid this, requests are asynchronous and need an identifier to tell the DALPHI framework to which request an answer belongs.

2.2.2. Named Entity Recognition Using NLTK

The assistance web service was developed using the Python programming language.[15] The Python community offers an exhaustive set of packages, and a noticeable amount of NLP libraries.[16] In this case NLTK's Maximum Entropy Classifier was used to provide an NER [21] assistance. Generally, the features of the Named Entity chunker were used as recommended:[17]

1. *shape:* Which shape does the current token have? Is it made up of numbers, punctuation, uppercase characters, lowercase characters, mixedcase, other?
2. *wordlen:* The number of characters of the token.
3. *prefix3:* The first three chars of the token.
4. *suffix3:* The last three chars of the token.
5. *pos:* The Part Of Speech (POS) tag, like Noun, Verb, Pronoun, Preposition, ...
6. *word:* The word itself.
7. *en-wordlist:* Is the word included in a english dictionary?
8. *prevtag:* The named entity tag of the previous token.
9. *prevpos:* The POS tag of the previous token.
10. *nextpos:* The POS tag of the next token.
11. *prevword:* The previous word.
12. *nextword:* The next word.
13. *wordnextpos:* The current token (lowercased), concatenated with the POS tag of the following token.
14. *posprevtag:* The POS tag of the current token, concatenated with the named entity tag of the previous token.
15. *shapeprevtag:* The shape of the current token, concatenated with the named entity tag of the previous token.

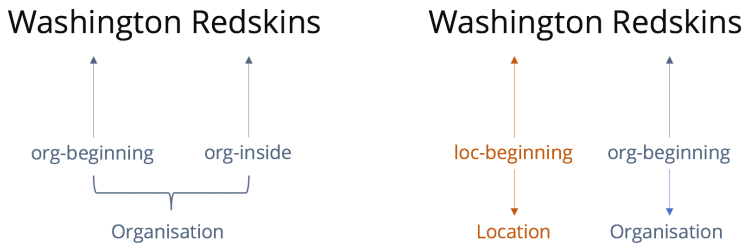
Adjustments for German Texts The following adjustments to the feature set were necessary to be able to use the chunker with German texts:

[15] Python 3.6.1: `https://www.python.org/downloads/release/python-361/` (last date of retrieval: August 11, 2017)
[16] Some of the better known ons are: NLTK, TextBlob, Stanford CoreNLP, ...
[17] See NLTK's documentation of the *nltk.chunk.named_entity* class: `http://www.nltk.org/_modules/nltk/chunk/named_entity.html` (last date of retrieval: August 11, 2017)

1. A German POS tagger was used. NLTK's default provides POS tagging for English sentences, which can not be applied to German texts. A suitable tagger was available as open source software via GitHub [22].
2. A German word list was employed. It is provided as open source as well and was created for the GNU Aspell project. [23]

This work is about the effect an AL based assistance system has on the performance of human annotators. The working assistance system with an underlying NER algorithm is a proof of concept. It shows that the general idea of employing AL as an automated annotation assistance actually works. Therefore, the classifier we chose – despite the availability of more performant alternatives [24] – is a solid, well known (due to its frequent use in other works), as well as easily comprehensible algorithm. It serves as a baseline of what is possible in the field of NER.

Mechanics of the Maximum Entropy Classifier The NER assistance system was implemented using a Maximum Entropy Classifier (MEC). Maximum Entropy modeling was first described by E. T. Jaynes in 1957 [25]. The classifier chooses from a set of labels and assigns each word one of it. Usually the labels are the different entities (person name, organization name, city, geo political entity, ...) concatenated with *beginning* or *inside* to describe each state a word can have [15]. For example, Figure 9 shows the term "Washington Redskins", describing an organization named "Washington Redskins" or a location "Washington" and a following organization "Redskins".



(a) Both words together are the name of an organization.

(b) Each word has a meaning of its own: A location, followed by an organization.

Figure 9: A word pair example with an ambiguous meaning.

Only the *beginning* or *inside* suffix of the labels make this clear: Either the labels *organization-beginning* followed by *organization-inside* are used and represent the first example interpretation ("Washington Redskins"), or the labels *location-beginning* followed by *organization-beginning* are used to represent the second interpretation. This way, the beginning / inside suffixes are used for disambiguation. Additionally, an *outside* tag is available to tag words not belonging to any entity. Figure 10 provides an example of a complete tagged sentence using this technique.

Figure 10: A named entity tagged example sentence using the beginning / inside / outside notation.

The maximum entropy classifier builds a model to predict the probability of a label y given an input word x, $P(y \mid x)$. It is used to answer the question "Which label suits a given input best?".[18] Not every label can always be applied: If the previous word was the beginning of a person's name the current word can not be tagged *location_inside* since this would require a preceding *location_beginning*. Instead, in this case, only the tags *person_inside*, *outside* or all beginning-tags would be possible.

Several features can be computed for every input word x, e.g. some binary ones like *is the word capitalized?* or *is the word included in a dictionary of the same language?*. Nominal variables may be included in the feature set as well, like a word's *Part Of Speech (POS) tag* (the word category like noun, verb, pronoun, ...) or even *the number of characters the word has*. These nominal variables will be converted to vectors of binary values before being applied to the algorithm [26]. This conversion results in relatively large but sparse binary *feature vectors* (sets of all features of an input word).

Features are represented in functions that can contain logic like the lookup in a dictionary. $g_i(x, y)$ is an example of such a feature function for an input word x:

$$g_i(x, y) = \begin{cases} 1, & x \notin german\_dictionary \wedge y = \text{org-beginning} \\ 0 & \text{otherwise} \end{cases} .$$

Here i indicates the number of the feature and *german_dictionary* in this example is a list of German words. A word that is not included in such a dictionary will most likely be a new word creation and hence it is a good candidate for a company name.

The model now computes α_i to the power of g_i and multiplies all these exponential terms. This way only those parameters which belong to a feature that results in 1 (or *true*) have an influence on the probability of a label (since all other terms $\alpha_i^0 = 1$ do not change the arithmetic product).

[18] Since this classifier belongs to the class of conditional classifiers it can also be used to ask for the probability of a given label for a given input; but in this case we are interested in the most likely label for each word of the plain, unannotated text.

Given an input word x, the classifier now computes the probability for every label y and choses the one with the highest $P(y \mid x)$. This term is defined as follows [27]:

$$P(y \mid x) = \frac{1}{Z(x,y)} \prod_{i=1}^{k} \alpha_i^{g_i(x,y)}.$$

With k being the number of features and $Z(x)$ being a normalization function to be able to sum up the probabilities of all the different possible labels to 1:

$$Z(x,y) = \sum_{y} \prod_{i} \alpha_i^{g_i(x,y)}.$$

The procedure of computing probabilities for every label to find the most likely one is repeated for every word of the corpus.

The α_i parameter plays probably the most important part in the maximum entropy approach. To compute these parameters, the model starts with randomly guessing them. In a next step it uses iterative search techniques to improve them, maximizing the total likelihood of a training corpus as follows [21]:

$$P(x) = \sum_{j \in C} P(y_j \mid x_j)$$

With C being the corpus of training data and y_j and x_j the label and the feature vector of the word j.

These search techniques are iteratively optimizing the parameter set. For this optimization we used *MegaM* [28], [29] in our implementation.[19] Other algorithms like the Generalized Iterative Scaling or Improved Iterative Scaling are suitable as well – but a lot slower [21]. This software employs the *limited-memory Broyden–Fletcher–Goldfarb–Shanno* algorithm [31] for parameter optimization of multinominal features like the ones we use in the NER case.[20] Like the original Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm it approximates an inverse Hessian matrix. The *limited memory* alternative of the BFGS algorithm does not store a dense estimation of this invers but only a sparse representation, which accelerates the computation.

[19] *MEGA Model Optimization Package* from the University of Utah: `https://www.aclweb.org/aclwiki/index.php?title=MegaM:_Maximum_Entropy_Model_Optimization_Package_(Repository)` (last date of retrieval: August 11, 2017). This parameter optimizer is recommended by NLTK's Maximum Entropy chunker documentation [30].

[20] The optimizer, as well as the maximum entropy model, could also be used for binomial variables. In this case the optimizer would use the *conjugate gradient* method [32].

Performance Evaluation To evaluate the performance of the system we implemented, the usual metrics precision, recall and F1-score were computed [33], [34]. These measurements are typically used to evaluate a binary classification system. In our case we have multiple possible labels per word. Here these metrics refer to each of the labels.

We assume for example that we want to annotate a word with one of two possible labels, A or B. If label A is correct for this word and a classifier assigned this label as well, we would call this a *true positive* (TP) annotation. But if this example classifier would instead label the test word with label B, this assignment would be a *false positive* FP and the missing label A would count as a *false negative* FN.

The following formulas use these abbreviations as quantities: TP for example stands for the number of *true positive* annotations.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

We created a maximum entropy model as we used for the assistance system and trained it to use the GermEval 2014 NER training data set [35] which contains 23.999 sentences and 29.076 annotations. The sentences were gathered from German Wikipedia articles and online news and covered the labels “Location”, “Person”, “Organization”, and “Other”.

We shuffled the data set and performed a 10-fold cross-validation. The arithmetic average of the 10 sub-evaluations was then calculated for the scores listed in Table 1. The scoring was done using NLTK’s evaluation function for sequence taggers.[21]

Comparing our results to the GermEval 2014 rankings [24], our model’s performance is in the lower quarter of the leaderboard. If we had participated in the competition with this model, our model would have defeated three of the eleven competitors.

To study if the general approach of an assistance system as described in this chapter helps to improve text annotations, we’re going to introduce the empirical method we used in the following chapter.

[21] `http://www.nltk.org/api/nltk.tag.html#nltk.tag.api.TaggerI.evaluate` (last date of retrieval: August 11, 2017)

Table 1: Scores of the Maximum Entropy Classifier model evaluation, compared with results from the GermEval 2014 (GE) competition [24].

Measure	Our model	GE lowest	GE median	GE highest
Precision	67.63	40.14	76.71	78.07
Recall	53.62	34.71	63.25	74.75
F1-Score	59.81	37.23	69.33	76.38

2.3. Simulation of the Assistance System for the Study

We tested three different levels of assistance. These differed in their level of recall; in other words: how many completely correct (label and span) pre-annotations they provided. To compare how pre-annotations with different recalls impact the human annotation performance, we compared systems with 10% correct pre-annotations, one with 50% correct pre-annotations and with 90% correct pre-annotations. 10% is quiet easily reachable, even with a simple model. This level represents the lower bound of what we assumed might be a useful assistance. We were mostly interested if we can find an effect of such an assistance, even if it does not provide sophisticated pre-annotations at all. The next system with a recall of 50.0% can realistically be created (this is less than the median of the performance of the GermEval competition [24]) and hence the most relevant of our comparison. Its suggestions are reasonable and we hoped to find a significant effect with this level of assistance. The last system, with 90% correct pre-annotations, is currently out of reach for a German NER (the recall of the best system of the GermEval competition was below 80.0% [24]). Here we created an upper bound of the possible and were interested in how this system influences the human performance, especially the perception of the annotators regarding workload and monotony.

The performance of the assistance system within one level should not vary at all, it should make comparable suggestions on the same level (in terms of correctness) for all participants. Furthermore, subjects should not need to train the assistance to the desired level of correctness due to time restrictions. Therefore the study was conducted with a simulation of an assistance system. The simulated assistance made no real suggestions, but it used a lookup table that predefined for every subject which words should be chunked and which label a chunk should have. The lookup table was balanced; every participant had a unique combination of chunks and labels. The subjects completed the annotation of all the texts that were predefined for them. This way, the impact of order was controlled. Also the distribution of the errors the simulation made was controlled.

If the assistance made only 10% correct suggestions, at the same time the remaining 90% were not correctly annotated. It follows that we had to define what *not correct* annotations are. We identified five possible types of error, ranging from not annotated

Table 2: Possible annotation errors. Green labels are person names, blue labels are company names.

Label	Span	Description	Example	Dist.
✓	✓	correct	CEO Lorene Duck raises wages.	57.8%
✓	✗	correct label, wrong span	CEO Lorene Duck raises wages.	7.8%
✗	✓	wrong label, correct span	CEO Lorene Duck raises wages.	3.1%
✗	✗	wrong label and span	CEO Lorene Duck raises wages.	0.0%
✗	✗	unnecessary annotation	CEO Lorene Duck raises wages.	1.5%
-	-	missing annotation	CEO Lorene Duck raises wages.	29.8%

at all to wrong annotated in terms of label and span. Both the assistance system and a human can make such mistakes. They result as a combination of mistaking the label of an annotation or the span. And since the assistance's output is exactly the same as what a human does by annotation (defining a chunk of words and label it), these mistake types apply for both of them. See Table 2 for an overview of possible annotation errors and a correct annotation as a reference.

To make the simulation as ecologically valid as possible, we analyzed the distribution of the errors mentioned in Table 2 using a state of the art Conditional Random Field (CRF) based NER model [36]. For this analysis we let the model predict the labels for the same corpus we used for the study. The *Dist.* column in Table 2 shows the distribution of mistakes the model made. We then applied the same distribution of errors to the NER simulation we used in the study. This way we ensured, that the frequency of errors exposed to the study subjects was relatively similar to real world errors.

The predefined annotations were made using a *gold standard* template which we hand-crafted on our own.[22] First, the distribution of errors for every assistance level was computed. Then this distribution was $\frac{N}{3}$ times shuffled with N being the total number of subjects. The resulting table defined for all N participants which annotation should be correct or wrong (and if it should be wrong, which error it should be). Finally this lookup table was used to generate artificial mistakes in the data.

Evaluation of the Simulated Assistance To evaluate the pre-annotations we generated, we computed precision, recall and F1-score measures of our simulated assistance system. We also calculated the same measures for our MEC model (trained on the GermEval training corpus as described for the performance evaluation in Section 2.2.2) to evaluate its performance on the study corpus (see Table 3).

[22] A gold standard in ML is an annotated data set that is considered as perfectly annotated as possible.

Our own model did not reach a recall of 50.0%. It is noticeable that this model was trained on a different corpus (possibly containing different text domains) and that this corpus was annotated with different guidelines than what we used for our gold standard. We show this result to demonstrate how less flexible models can be in terms of using them in different contexts – and to highlight the importance of a rapid training data generation tool, to be able to create models for different use cases (like following different annotation guidelines). Nevertheless, the GermEval competition shows that a recall of at least 50.0% is definitely feasible (as seen in Table 1).

Table 3: Evaluation of the simulated assistances and our Maximum Entropy Classifier model.

Measure	10% assistance	50% assistance	90% assistance	Our model
Precision	26.96	78.00	96.54	74.69
Recall	10.0	50.32	90.0	39.03
F1-Score	14.59	61.18	93.16	51.27

3. Method

To understand the impact of pre-annotated text on the performance of human annotators, to validate Day's [9] findings of increased productivity using pre-annotations and to estimate whether the development of such a pre-annotation system will pay off, a comprehensive study was conducted. For this, the assistance system was simulated and statistically tested using the example task of annotating data for NER. Furthermore we discuss mistakes the assistance makes and what annotation mistakes of machine and human look like.

3.1. Sample

To receive insightful results we computed a statistical power analysis using the *G* Power 3* [37] tool.[23] Since we planned to conduct our main results using an ANOVA, we used the *ANOVA: Fixed effects, omnibus, one-way* test with the following parameters:

- Effect size f: 0.4 (large effect cf. Cohen [38])
- α: 0.05
- Power $(1 - \beta)$: 0.8
- Number of Groups: 3

This analysis resulted in $n \geq 66$ participants. We recruited participants using the "probandenportal" from the *Technische Universität Berlin* as well as via personal invitations.[24] All participants were tested in lab conditions to control outside influences; they all were tested during daytime with natural light from the windows in the same, quiet environment on equal instruments and under equal conditions.

All participants spoke German fluently and none suffered from color blindness of any kind.

In the beginning of the experiment we incentivized with the chance to win one of two shopping vouchers in the lottery between all the participants or to receive one "VP Stunde" as a reward for their participation.[25] Since we did not find enough participants we changed the reward to 10 € for the 20 last participants.

[23] The freeware *G* Power 3* can be downloaded from `http://www.gpower.hhu.de/` (last date of retrieval: August 11, 2017).

[24] The "probandenportal" from the *Technische Universität Berlin*: `https://proband.ipa.tu-berlin.de/` (last date of retrieval: August 11, 2017)

[25] It is common practice at the chair where the study was conducted to offer "VP stunden" to participating students as a motivation. All students at this chair are required to collect at least ten "VP stunden" during their degree course.

3.2. Apparatus

All participants conducted the study on a laptop (MacBook Air 6.1 with an 11 inch display and a resolution of 1366 × 768 pixels). A mouse on a mousepad was connected to the laptop to be used by the participants. The documents for the study were a single page of terms and conditions as well as a short description of the task, a double page document of annotation guidelines and a single page document describing functions of the interface they were going to use (see Appendix E).

3.2.1. Annotation Interface

The tool the participants used was the annotation interface for sequence labeling of the DALPHI annotation framework (see Appendix B for a link to the software itself). This interface is a web app that ran in full screen mode, see Figure 11 for an exemplary screenshot.

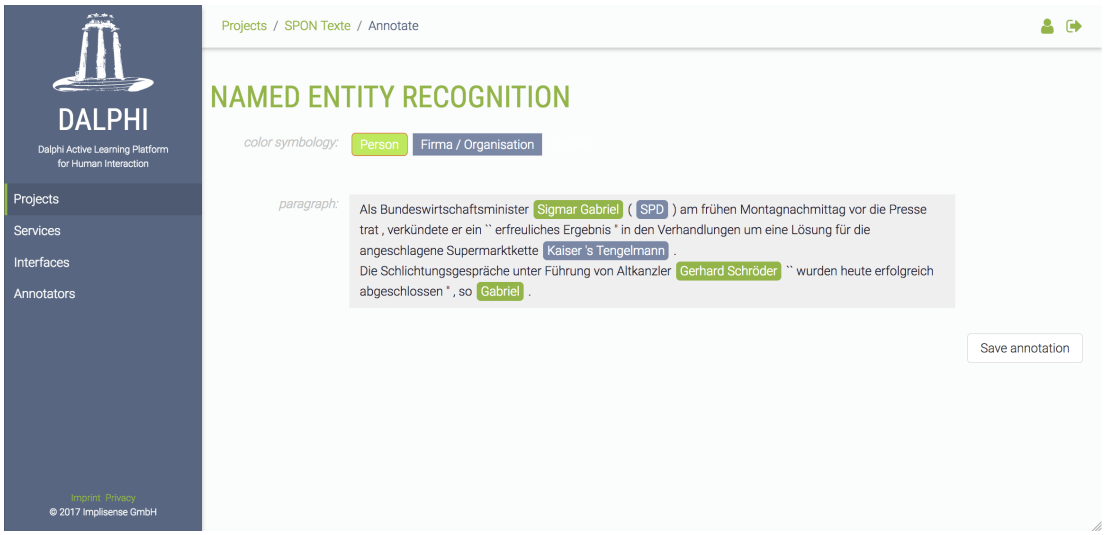


Figure 11: The DALPHI UI for text annotations.

All the UI elements the participants could use are the two example labels above the grey paragraph container to select the label (category) of the current selected annotation, the container itself and the "save annotation" button below the container. Within the container, every word was clickable to create a new annotation. Several interactions with annotations and the mouse or the keyboard were possible to modify annotations (change span or label or to remove it), the document "Funktionen des Annotationsinterfaces" (see Appendix E) that was available to the participants as well, provides a detailed description of all the functions of the UI.

The text container itself displayed one out of 73 paragraphs of German running text (with 305 sentences and approximately 6000 words in total). After one paragraph was

processed, the annotations were saved and the next one was displayed. The text to work with consisted of 14 recent news articles from different sources, chosen by total length and count of possible annotations (original texts in Appendix C). We compiled the collection to be as diverse as possible in order to mute performance differences of the participants with different expertise and affinity to different subjects.

3.2.2. Assistance System

The assistance we used for the study was simulated as described in Section 2.3. The study phase was divided into four blocks of approximately equal amounts of annotations (see Figure 12). The assistance was presented in two out of four blocks which we discuss in more detail in Section 3.5.

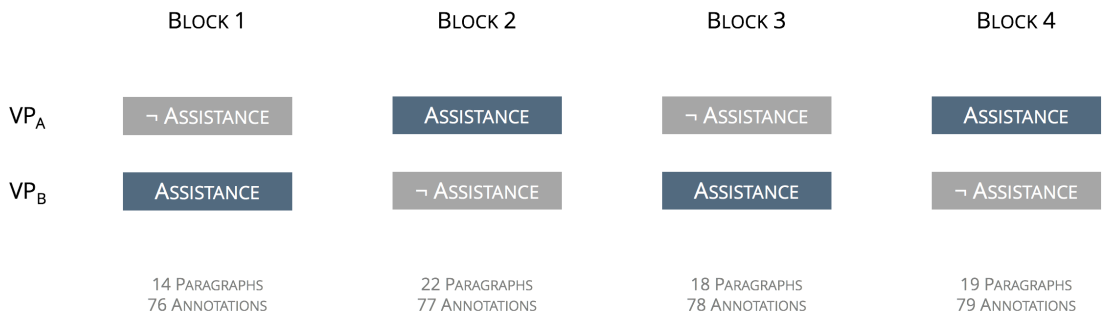


Figure 12: Four blocks of the study, the assistance's presence was toggled.

According to gold standard there were 310 spots to annotate in the entire text. To distribute the annotations as equally as possible between the four blocks, we put between 76 and 79 correct annotations into each block. Each block had to contain an entire paragraph. Since we used real texts we couldn't control the number of annotations per paragraph perfectly, which explains the small differences between the number of annotations per block.

3.2.3. Questionnaires

After each block, the participants were asked to fill in a questionnaire about how they felt in terms of perceived workload and monotony of the task. Additionally we asked them how they would estimate the reliability and the correctness of the assistance. These questionnaires were embedded into the UI the participants worked with; after a block of annotations the questionnaires appeared on the same screen. See Figure 21 (in Appendix D) for a screenshot of the questionnaire.

Lastly, after all annotation blocks and their related questionnaires were completed, the participants were asked to fill in a demographic questionnaire (see Figure 22, Appendix D).

3.3. Procedure

The procedure of the study was divided into three phases: Participants started with the preparation phase, made their own experiences with the annotation UI in the training phase and annotated their texts alone in the experimental phase.

Preparation Phase The instructor sat next to the participants during the three preparation steps and participants were encouraged to ask questions right away.

1. Participants received a document that described the following study and gave them a general idea what it was about and why this was interesting. The same sheet contained a section about their consent to participate, which they had to sign before proceeding (see Appendix E.1).
2. The participants then read the annotation guidelines. These should help whenever the participants were not sure how to annotate something and explained their task with several examples. This document was available to the participants for the whole time of the study so that they could take a look at it whenever they wanted to (see Appendix E.2).
3. Finally, the participants read a document about the interface they were going to use. All of the interface's functions were explained in the document (see Appendix E.3). While they read it the instructor presented each of the explained functionalities using the interface.

Training Phase After the participants were finished, the instructor asked them to try out each of the described functions on their own to make sure every part of the interface was well understood. For this they were asked to complete a task for each of the interface's functions. While doing so they were allowed to read the guidelines and the interface's description. Additionally they could ask for help.

Experimental Phase The participants were asked to annotate person and organization names in the whole corpus using the interface described in Section 3.2.1. Participants were instructed that their goal was to complete all texts. There was no time restriction. Additionally, the participants were asked to fill in the questionnaires as described in Section 3.2.3. The annotation, as well as the questionnaires, were processed within the DALPHI framework; no contextual change was required.

Participants completed the study itself on their own and without having the instructor sit next to them. Instead, the instructor sat in the same room, but several meters away and without facing the participants in order to avoid distractions and to not convey the feeling of being observed. They were allowed and encouraged to ask for help whenever they needed it. If a question was about a technical detail the instructor helped and answered distinctly; if the question was about what to do or how to annotate a certain

sentence, the instructor replied with an example from the guidelines and asked if the subject could find a similar case that is described in the guidelines on their own. The instructor never revealed the direct solution.

3.4. Data Analysis

After the participants annotated their texts, the results were 66 different annotated but textually identical corpora. We created a data preprocessing pipeline to match each annotated text with the gold standard. This way we could quantify which annotations were correct, missing or of one of the error types described in Table 2. As a result, this pipeline generated a data frame we used for our statistical analysis.

For our hypothesis tests (see Section 3.6) we employed one sample T-Tests and ANOVA. Theses tests make the assumptions of normal distributed data and of homogeneity of variances [39].[26] To test if these assumptions hold, we used the *Shapiro-Wilk test* [40] to test a data (sub-)set for normal distribution and *Levene's test* [41] to test if the assumption of homogeneity of variances holds for a given data (sub-)set.

If the assumptions of the tests were not met, we used non-parametric alternative tests that do not rely on these assumptions. In the case of a not normally distributed data set, we used *Wilcox' Robust Statistics*, which uses a one-way comparison of multiple trimmed group means [42]. If the homogeneity of variances was violated, *Welch's F* [43] was employed.

To contrast two groups against each other, if an ANOVA Test is significant, two T-tests were conducted to discriminate which of the three different assistance levels were accountable for the significant impact. Since all our hypothesis were based on the assumption "the better the assistance, the better the human performance", only two out of the three possible comparisons were made. Namely 10% vs. 50% and 50% vs. 90%, but never 10% vs. 90%. For that we used the *Bonferroni adjustment* [44] to decrease our α-level to 0.025 since we conducted only two post-hoc tests.

3.5. Experimental Design and Variables

The study was built as a 3×2 mixed design: The corpus was equal for all participants. It differed in the pre-annotations it had; these were balanced between the participants. Each participant was assigned to one of the three different levels of the assistance (10%, 50% and 90% correct pre-annotations). In addition to that, the assistance system was alternating between being present and not being present (two out of four blocks each). If a participant started with the assistance, the following block would be without it and vice versa. We toggled whether a participant started with the assistance in the first block

[26] Besides other assumptions like a proper scale of measurement or a random sample.

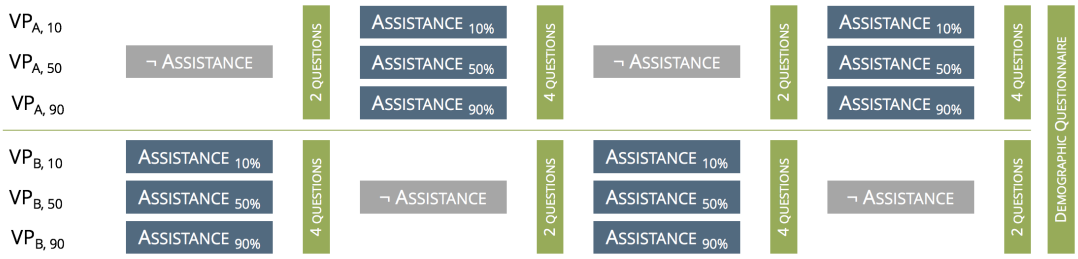


Figure 13: The phases of the study for all groups.

or not, so we ended up with half of the participants having started with the assistance system, and the other without it. This factor was used for balancing purposes only.

To measure only the impact of the assistance system but not the individual performance we condensed these four blocks computing the difference between a block with assistance minus the previous or following block without assistance. As a result, we obtained two measurement points of time, a first one and a second one; each describing a relative change leading back to the assistance system. Figure 13 depicts this structure, showing all three different groups of assistance levels, measurement points of time and when the questionnaires were shown.

Independent Variables The described 3×2 mixed design contains two independent variables: level of assistance and point of time measurement. The level of the assistance system has three distinctions: 10%, 50% and 90%. This was operationalized through the simulated pre-annotation process using the gold standard template method described in Section 2.3. The second variable is the point of time measurement, namely the first half or the second half of the study (both halves combine the difference of two blocks, so two halves contain results from all four blocks). A third independent variable, indicating whether a participant started with or without the assistance, was a control variable which we balanced. It will not be considered any further.

Dependent Variables As described in Section 1.3, we were interested in examining the impact of the assistance regarding the annotation efficiency. We identified three relevant performance dimensions: **Correctness** (how many annotations have been annotated correctly?), **misses** (how many annotations have been missed?) and **tempo** (what ist the average time spent per correct annotation?). Therefore we were interested in analyzing how these parameters were affected by the assistance system. The differences we computed between two blocks were percentages in the case of correctness and misses, and seconds in the case of tempo.

Operationalization One dependent variable was the processing time of a whole paragraph in seconds (for technical reasons we could not measure the processing time of each annotation individually). To derive the processing time of a single, correct annotation (in which we were actually interested in), we computed the average processing time by summarizing the processing time of all paragraphs per block and divided it using the total count of correct annotations of the same block.

The other dependent variable was the set of all the annotations each participant created. From these sets of annotations we derived two more meaningful variables that the plain annotations themselves constitute: The percentage of correct annotations and the percentage of missed annotations. We obtained these values using the gold standard matching technique and computed the two desired percentages for each block.

As already mentioned, we then computed the differences between assistance and no assistance of all of these three per-block-values. We subtracted each of the values of a block with assistance minus one block without assistance. We did so for the first, as well as the second half. This way we compared only the influence the assistance had on the participants, rather than individual performance differences.
For the following hypotheses A, B and C we computed the average of the differences of the first and the second half per participant. For these hypotheses we were just interested in a general influence of the assistance system, therefore we combined the data from the two halves.

3.6. Hypotheses

Generally, we expected that the assistance system would increase an annotator's performance in every of the three performance dimensions. Here we just list the alternative hypotheses; the corresponding $H0$s (the assistance system does not increase respectively decrease a performance dimension) are given implicitly.

Regarding the **correctness** we therefore formulated the following three hypotheses:

Ai) A 10% correct assistance system increases the correctness of annotations compared to annotations done without the assistance.

Aii) A 50% correct assistance system increases the correctness of annotations compared to annotations done without the assistance.

Aiii) A 90% correct assistance system increases the correctness of annotations compared to annotations done without the assistance.

Regarding the **tempo** we formulated the following three hypotheses:

Bi) A 10% correct assistance system decreases the average time needed for one annotation, compared to annotations done without the assistance.

Bii) A 50% correct assistance system decreases the average time needed for one annotation, compared to annotations done without the assistance.

Biii) A 90% correct assistance system decreases the average time needed for one annotation, compared to annotations done without the assistance.

And regarding the **misses** we formulated the following three hypotheses:

Ci) A 10% correct assistance system decreases the miss rate per block, compared to annotating a block of text done without the assistance.

Cii) A 50% correct assistance system decreases the miss rate per block, compared to annotating a block of text done without the assistance.

Ciii) A 90% correct assistance system decreases the miss rate per block, compared to annotating a block of text done without the assistance.

Finally we wanted to **compare the impact of the different levels** of the assistance system to each other. Therefore we formulated the following hypotheses:

Di) The 50% correct assistance system will increase the influence it has on correctness, compared to the 10% correct assistance system.

Dii) The 90% correct assistance system will increase the influence it has on correctness, compared to the 50% correct assistance system.

Ei) The 50% correct assistance system will increase the influence it has on tempo, compared to the 10% correct assistance system.

Eii) The 90% correct assistance system will increase the influence it has on tempo, compared to the 50% correct assistance system.

Fi) The 50% correct assistance system will increase the influence it has on the miss rate, compared to the 10% correct assistance system.

Fii) The 90% correct assistance system will increase the influence it has on the miss rate, compared to the 50% correct assistance system.

For a short summary of the introduced hypotheses, Table 4 provides an overview.

Table 4: Summary of the hypotheses

	correctness	tempo	misses
10% correct Assistance	increase (Ai)	decrease (Bi)	decrease (Ci)
50% correct Assistance	increase (Aii)	decrease (Bii)	decrease (Cii)
90% correct Assistance	increase (Aiii)	decrease (Biii)	decrease (Ciii)
10% vs. 50%	increase (Di)	decrease (Ei)	decrease (Fi)
50% vs. 90%	increase (Dii)	decrease (Eii)	decrease (Fii)

4. Results

We are going to describe the results of our study in detail. We obtained significant results for all our hypotheses testing the assistance with 50% or 90% correct pre-annotations. The tests of the assistance with only 10% correct pre-annotations were not significant (Section 4.1). Neither are the comparisons of different assistance levels (Section 4.2).

4.1. Impact of the Assistance System on the Three Performance Parameters

This section presents our analysis results of the t-tests we conducted for the hypotheses groups A, B and C. All tests in this section used a significance level of $\alpha = 0.05$.

4.1.1. Impact of the Assistance on Correctness

Figure 14 shows a comparison of the three different assistance levels regarding the correctness of the participants' annotations. The level with 10% correct pre-annotations was not significantly greater than zero, whereas the level 50% and the level 90% assistance revealed significant results.

Hypothesis Ai "A 10% correct assistance system increases the correctness of annotations compared to annotations done without the assistance."

There was no significant difference, $t(21) = -0.521, p = 0.608$ ($M = -0.62, SD = 5.601$), so the null hypothesis is not rejected.

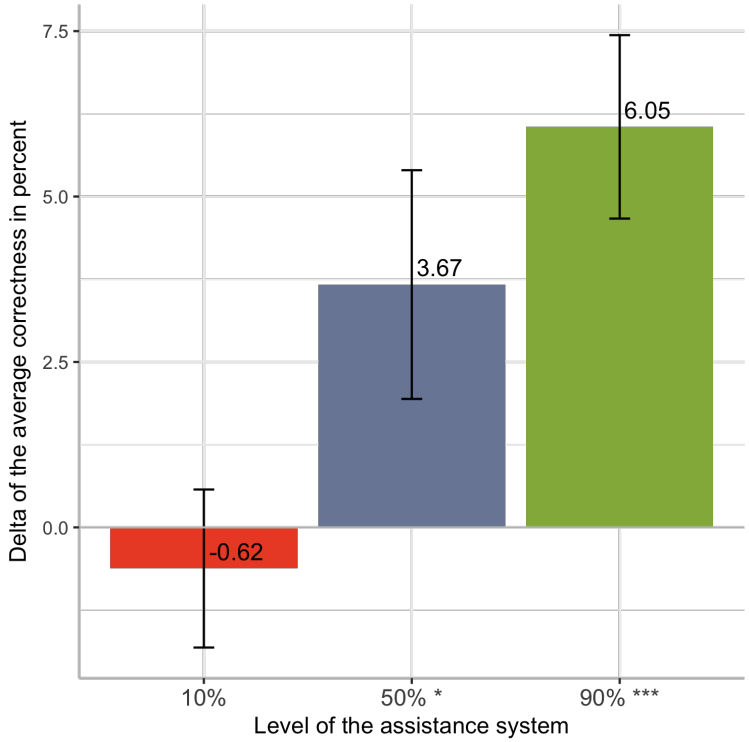


Figure 14: All three levels of assistance regarding the performance dimension correctness. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

Hypothesis Aii "A 50% correct assistance system increases the correctness of annotations compared to annotations done without the assistance."

There was a significant difference, $t(21) = 2.123, p = 0.046$ ($M = 3.67, SD = 8.107$), so the null hypothesis is rejected at the given level of significance.

Hypothesis Aiii "A 90% correct assistance system increases the correctness of annotations compared to annotations done without the assistance."

There was a significant difference, $t(21) = 4.367, p < 0.001$ ($M = 6.05, SD = 6.502$), so the null hypothesis is rejected at the given level of significance.

4.1.2. Impact of the Assistance on Tempo

Figure 15 shows a comparison of the three different assistance levels regarding the tempo of the participants. The level with 10% correct pre-annotations was not significantly less than zero, whereas the level 50% and the level 90% assistance revealed significant results.

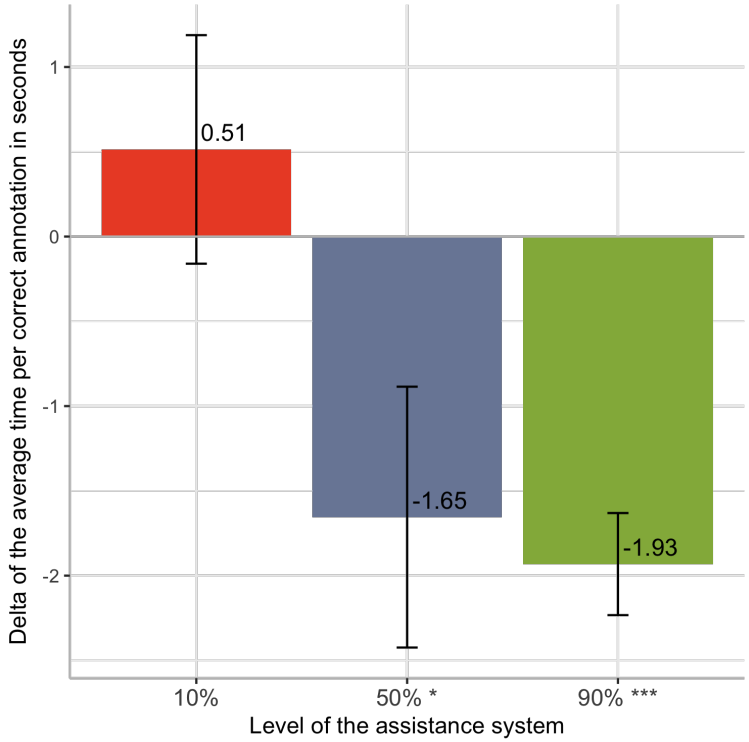


Figure 15: All three levels of assistance regarding the performance dimension correctness. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

Hypothesis Bi "A 10% correct assistance system decreases the average time needed for one annotation compared to annotations done without the assistance."

There was no significant difference, $t(21) = 0.762, p = 0.454$ ($M = 0.51, SD = 3.161$), so the null hypothesis is not rejected.

Hypothesis Bii "A 50% correct assistance system decreases the average time needed for one annotation, compared to annotations done without the assistance."

There was a significant difference, $t(21) = -2.151, p = 0.043$ ($M = -1.65, SD = 3.608$), so the null hypothesis is rejected at the given level of significance.

Hypothesis Biii "A 90% correct assistance system decreases the average time needed for one annotation compared to annotations done without the assistance."

There was a significant difference, $t(21) = -6.416, p < 0.001$ ($M = -1.93, SD = 1.412$), so the null hypothesis is rejected at the given level of significance.

4.1.3. Impact of the Assistance on Misses

Figure 16 shows a comparison of the three different assistance levels regarding the miss rate of the participants. The level with 10% correct pre-annotations was not significantly less than zero, whereas the level 50% and the level 90% assistance revealed significant results.

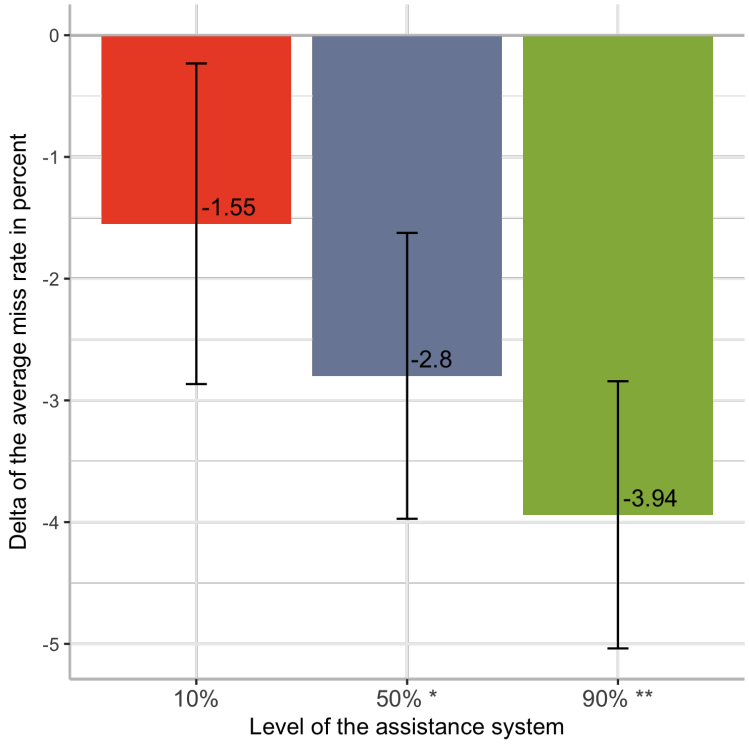


Figure 16: All three levels of assistance regarding the performance dimension misses. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

Hypothesis Ci "A 10% correct assistance system decreases the miss rate per block compared to annotating a block of text done without the assistance."

There was no significant difference, $t(21) = -1.176, p = 0.252$ ($M = -1.55, SD = 6.176$), so the null hypothesis is not rejected.

Hypothesis Cii "A 50% correct assistance system decreases the miss rate per block compared to annotating a block of text done without the assistance."

There was a significant difference, $t(21) = -2.383, p = 0.027$ ($M = -2.80, SD = 5.508$), so the null hypothesis is rejected at the given level of significance.

Hypothesis Ciii "A 90% correct assistance system decreases the miss rate per block compared to annotating a block of text done without the assistance."

There was a significant difference, $t(21) = -3.591, p = 0.002$ ($M = -3.94, SD = 5.147$), so the null hypothesis is rejected at the given level of significance.

4.2. Differences of the Three Levels of Assistance

For a further analysis of the data, involving the different points of time measurement and looking for main- and interaction effects, we conducted an analysis of variance. As predictor variables we used the level of the assistance and the point of time. If a difference between the levels of assistance can be detected, the change of the assistance level would have a main effect on the respective performance dimension.

The test of the hypothesis groups D, E and F showed no significant differences. Neither when looking at the influence of the assistant system with 10% correct suggestions compared to the system with 50% correct suggestions, nor when looking at the assistant system with 50% correct suggestions compared to the system with 90% correct suggestions.

4.2.1. Correctness

Computing an analysis of variance yielded in the results shown in Table 5.

Table 5: Results of the ANOVA, analyzing the assistance system's influence on correctness.

Effect	DFn	DFd	SSn	SSd	F	p	significance
Intercept	1	63	1214.7	5853.3	13.1	0.0006	✓
level	2	63	1006.8	5853.3	5.4	0.0067	✓
block	1	63	141.2	4535.8	2.0	0.1663	✗
level:block	2	63	125.7	4535.8	0.9	0.4226	✗

$\Rightarrow$ The p-value of the level of the assistance system is less than $\alpha = 0.05$; this result **is significant**; we found a main effect of the level of the assistance. Besides this there is no significant main effect of the block (the point of time measurement) and no significant interaction effect of the level and the block.

Hypothesis Di "The 50% correct assistance system will increase the influence it has on correctness, compared to the 10% correct assistance system."

We now carried out a contrast between assistance level 10% and 50% to validate our hypothesis Di. For this post-hoc test we adjusted the significance level to $\alpha = 0.025$. There was no significant difference, $t(37.328) = -2.042, p = 0.048$, so the null hypothesis is not rejected.

Hypothesis Dii "The 90% correct assistance system will increase the influence it has on correctness, compared to the 50% correct assistance system."

To evaluate the hypothesis Dii, we carried out a second contrast between assistance level 50% and 90%. For this post-hoc test we adjusted the significance level to $\alpha = 0.025$ as well. There was no significant difference, $t(40.109) = -1.076, p = 0.289$, so the null hypothesis is not rejected.

4.2.2. Tempo

Computing an analysis of variance got us in the results shown in Table 6.

Table 6: Results of the ANOVA, analyzing the assistance system's influence on tempo.

Effect	DFn	DFd	SSn	SSd	F	p	significance
Intercept	1	63	138.5	1050.3	8.3	0.0054	✓
level	2	63	157.8	1050.3	4.7	0.0122	✓
block	1	63	3.7	359.9	0.6	0.4237	✗
level:block	2	63	9.5	359.9	0.8	0.4394	✗

$\Rightarrow$ The p-value of the level of the assistance system is less than $\alpha = 0.05$; this result **is significant**; we found a main effect of the level of the assistance. Besides this there is no significant main effect of the block (the point of time measurement) and no significant interaction effect of the level and the block.

Hypothesis Ei "The 50% correct assistance system will increase the influence it has on tempo, compared to the 10% correct assistance system."

We carried out a contrast between assistance level 10% and 50% to validate our hypothesis Ei. For this post-hoc test we adjusted the significance level to $\alpha = 0.025$. There was no significant difference, $t(41.287) = 2.120, p = 0.040$, so the null hypothesis is not rejected.

Hypothesis Eii "The 90% correct assistance system will increase the influence it has on tempo, compared to the 50% correct assistance system."

To evaluate the hypothesis Eii, we carried out a second contrast between assistance level 50% and 90%. For this post-hoc test we adjusted the significance level to $\alpha = 0.025$ as well. There was no significant difference, $t(27.286) = 0.34, p = 0.740$, so the null hypothesis is not rejected.

4.2.3. Misses

Computing an analysis of variance produced in the results shown in Table 7.

Table 7: Results of the ANOVA, analyzing the assistance system's influence on the rate of missed annotations.

Effect	DFn	DFd	SSn	SSd	F	p	significance
Intercept	1	63	1007.3	3988.8	15.9	0.0002	✓
level	2	63	125.8	3988.8	1.0	0.3759	✗
block	1	63	152.6	2055.1	4.7	0.0343	✓
level:block	2	63	13.0	2055.1	0.2	0.8197	✗

$\Rightarrow$ The p-value of the block of the assistance system is less than $\alpha = 0.05$; this result **is significant**; we found a main effect of the point of time measurement. Besides this there is no significant main effect of the level of the assistance system and no significant interaction effect of the level and the block.

4.3. Questionnaires

The evaluation of the questionnaires was an explorative analysis since we did not have any hypothesis regarding the questions we asked. We processed the data in the same way as the data regarding the performance dimensions: We calculated differences of blocks with the assistance system and blocks without the assistance.

We showed that the assistance system can decrease the perceived workload of the annotators – if its suggestions are very accurate. Furthermore, we found that the perceived monotony of the annotators won't change, no matter what accuracy level the assistance has had or if it was present at all.

4.3.1. Perception of Workload

We started the evaluation of the questionnaires similar to how we proceeded with the performance dimensions. First, we conducted three t-tests for each of the assistance level in order to see whether there is a significant influence of the assistance system regarding the perceived workload of the participants. Figure 17 shows the perceived workload in respect to the three different assistance levels.

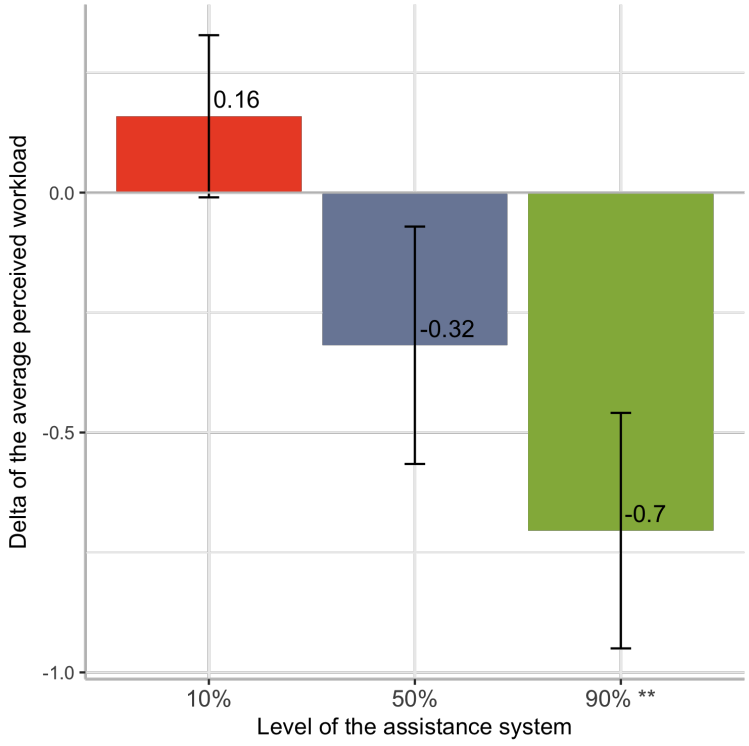


Figure 17: All three levels of assistance regarding the perceived workload of the subjects. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

Three Independent T-Tests For the following three tests we used a significance level of $\alpha = 0.05$. We starte with the evaluation of the workload differences of an assistance system with 10% correct suggestions and no assistance system. There was no significant difference, $t(21) = 0.941, p = 0.357$ ($M = 0.16, SD = 0.793$), so the null hypothesis is not rejected.

We continued with the evaluation of the workload differences of an assistance system with 50% correct suggestions and no assistance system. There was no significant difference, $t(21) = -1.286, p = 0.212$ ($M = -0.32, SD = 1.160$), so the null hypothesis is not rejected.

Finally we evaluated the workload differences of an assistance system with 90% correct suggestions and no assistance system. There was a significant difference, $t(21) = -2.870, p = 0.009$ ($M = -0.70, SD = 1.151$), so the null hypothesis is rejected at the given level of significance.

ANOVA for Main- and Interaction Effects Subsequently we computed an analysis of variance to find main- and interaction effects. Results are shown in Table 8.

Table 8: Results of the ANOVA, analyzing the perception of workload (via questionnaires).

Effect	DFn	DFd	SSn	SSd	F	p	significance
Intercept	1	63	10.9	138.6	5.0	0.0293	✓
level	2	63	16.5	138.6	3.7	0.0291	✓
block	1	63	0.5	58.4	0.5	0.4723	✗
level:block	2	63	1.1	58.4	0.6	0.5538	✗

$\Rightarrow$ The p-value of the level of the assistance system is less than $\alpha = 0.05$, this result **is significant**; we found a main effect of the level of assistance. Besides this there is no significant main effect of the block (the point of time of measurement) and no significant interaction effect of the level and the block.

4.3.2. Perception of Monotony

The second evaluation of the questionnaires was about the participant's perceived monotony. Again, we first conducted three t-tests to see whether there is a significant influence of the assistance system regarding the perceived monotony. Figure 18 shows the perceived monotony in respect to the three different assistance levels.

Three Independent T-Tests For the following three tests we used a significance level of $\alpha = 0.05$. We started with the evaluation of the monotony differences of an assistance system with 10% correct suggestions and no assistance system. There was no significant difference, $t(21) = 0.318, p = 0.754$ ($M = 0.05, SD = 0.671$), so the null hypothesis is not rejected.

We continued with the evaluation of the monotony differences of an assistance system with 50% correct suggestions and no assistance system. There was no significant difference, $t(21) = -1.405, p = 0.175$ ($M = -0.34, SD = 1.138$), so the null hypothesis is not rejected.

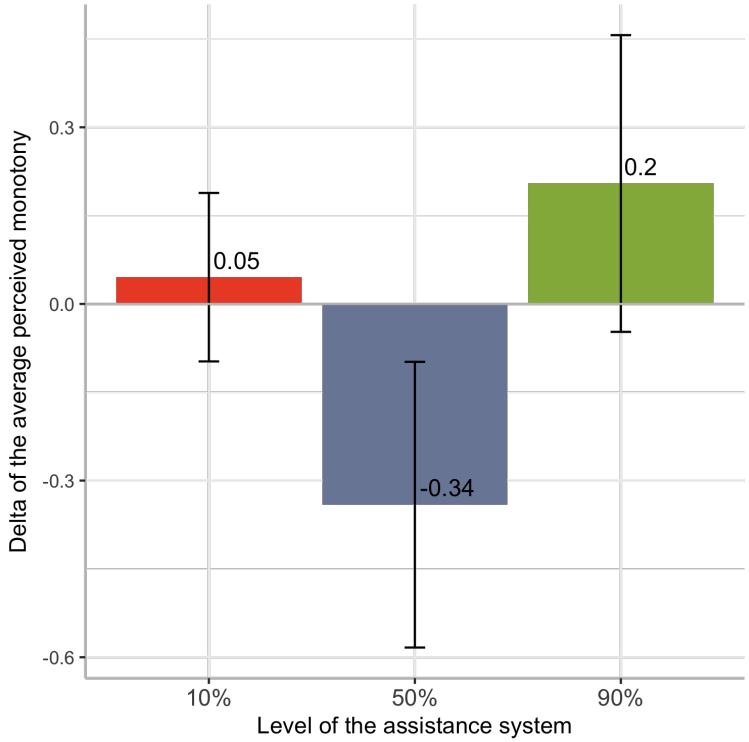


Figure 18: All three levels of assistance regarding the perceived monotony of the subjects. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

Finally we evaluated the monotony differences of an assistance system with 90% correct suggestions and no assistance system. There was no significant difference, $t(21) = 0.812, p = 0.426$ ($M = 0.20, SD = 1.182$), so the null hypothesis is not rejected.

4.3.3. Perception of Reliability

Figure 19 shows the perceived reliability in respect to the three different assistance levels.

ANOVA for Main- and Interaction Effects We computed an ANOVA to find main- and interaction effects. Results are shown in Table 9.

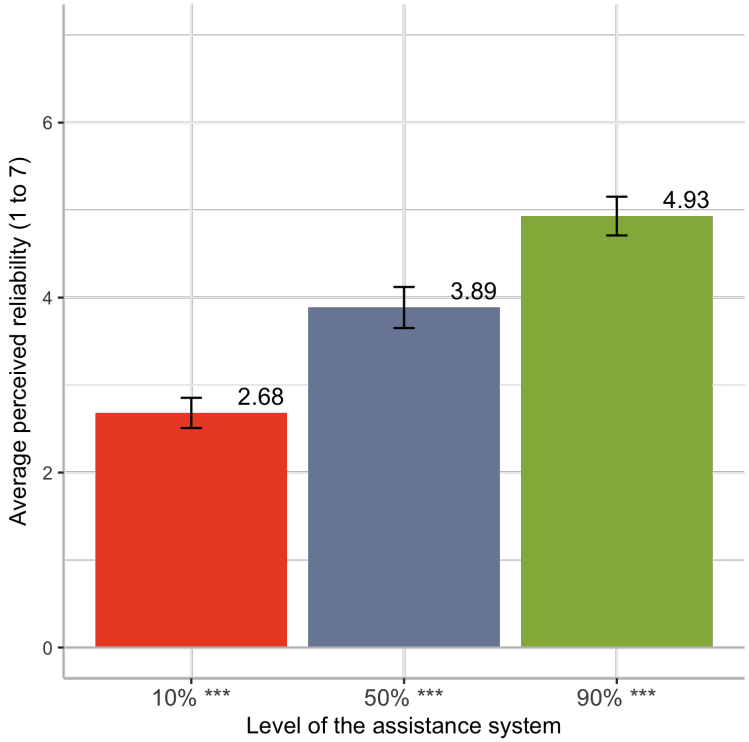


Figure 19: All three levels of assistance regarding the perceived reliability of the assistance system. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

Table 9: Results of the ANOVA, analyzing the perception of reliability (via questionnaires).

Effect	DFn	DFd	SSn	SSd	F	p	significance
Intercept	1	63	1939.7	123.8	987.285	< 0.001	✓
level	2	63	111.6	123.8	28.392	< 0.001	✓
block	1	63	3.0	66.0	2.895	0.09	✗
level:block	2	63	0.01	66.0	0.007	0.993	✗

$\Rightarrow$ The p-value of the level of the assistance system is less than our significance level of $\alpha = 0.05$, this result **is significant**; we found a main effect of the level of the assistance. Besides this there is no significant main effect of the block (the point of time of measurement) and no significant interaction effect of the level and the block.

4.3.4. Perception of Correctness

Figure 20 shows the perceived correctness in respect to the three different assistance levels.

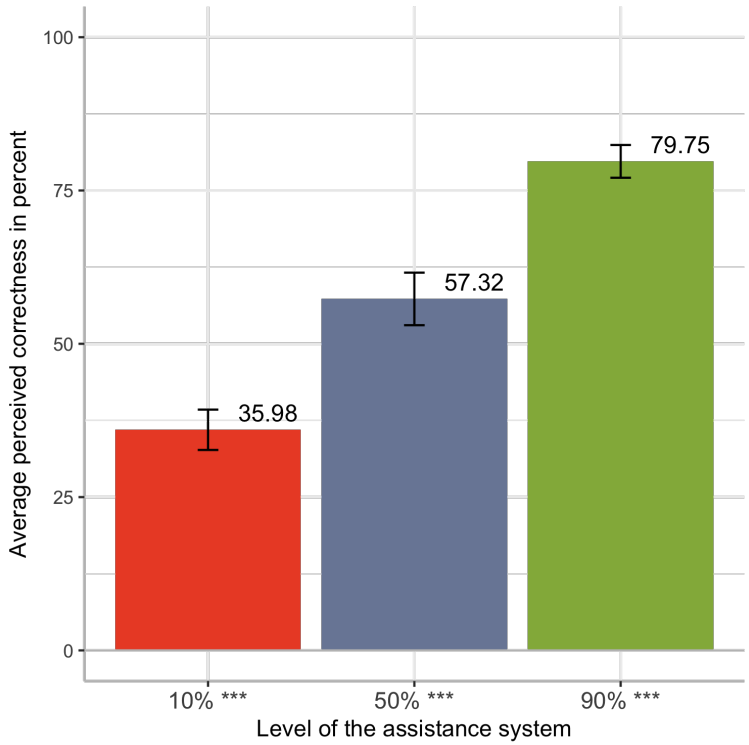


Figure 20: All three levels of assistance regarding the perceived correctness of the assistance system. The height of each bar is the mean value of the condition, the whiskers depict the standard error of the mean in both directions.

ANOVA for Main- and Interaction Effects We computed an ANOVA to find main- and interaction effects. Results are shown in Table 10.

Table 10: Results of the ANOVA, analyzing the perception of correctness (via questionnaires).

Effect	DFn	DFd	SSn	SSd	F	p	significance
Intercept	1	63	439189.4	33567.8	824.271	< 0.001	✓
level	2	63	42161.9	33567.8	39.565	< 0.001	✓
block	1	63	120.3	8321.7	0.911	0.344	✗
level:block	2	63	219.0	8321.7	0.829	0.441	✗

$\Rightarrow$ The p-value of the level of the assistance system is less than our significance level of $\alpha = 0.05$, this result **is significant**; we found a main effect of the level of the assistance. Besides this there is no significant main effect of the block (the point of time of measurement) and no significant interaction effect of the level and the block.

5. Discussion

Our research questions guided us to examine if and how it is possible to improve the task of manual data labeling. In particular, we wanted to know if it is possible to increase the performance of the annotators in terms of more correct, faster and more complete annotations. We clearly showed that this is possible – provided that the assistance has a certain level of correctness. We use our questionnaires in order to answer if we can improve the task itself: According to them we can reduce the perceived workload using the assistance; the system has to be really advanced, though, to have a significant impact on this dimension. However, the assistance did not change the perceived monotony at all. Thus, we could barely improve the task itself, from an annotator's perspective. Nevertheless, we achieved notable results in improving the outcome of text annotations and therefore its efficiency.

5.1. Hypotheses

We will discuss the results of our hypotheses tests in the same order as we introduced them.

Hypothesis Ai "A 10% correct assistance system increases the correctness of annotations compared to annotations done without the assistance."

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system did (statistically) not support the annotation task in terms of correctness, when it only made 10% correct (and 90% wrong) suggestions. The mean difference between blocks with assistance and blocks without is -0.62%. Since this value is negative, the assistance system with 10% correct suggestions did (on a descriptive level) impair the correctness that participants could achieve using this system.

Hypothesis Aii "A 50% correct assistance system increases the correctness of annotations compared to annotations done without the assistance."

$\Rightarrow p < \alpha$; this result **is significant**; the $H0$ is rejected. Here, the assistance system did (statistically) support the annotation task in terms of correctness, when it made 50% correct (and 50% wrong) suggestions. On average, the annotations were done about 3.7% more correctly with the assistance present than without its presence. This is an increase of the baseline from 83.93% to 87,63%.

Hypothesis Aiii "A 90% correct assistance system increases the correctness of annotations compared to annotations done without the assistance."

$\Rightarrow p < \alpha$; this result **is significant**; the $H0$ is rejected. Consequently the assistance system did (statistically) support the annotation task in terms of correctness, when it made 90% correct (and 10% wrong) suggestions. On average, the annotations were done about 6.1% more correctly with the assistance present than without its presence. This is an increase of the baseline from 83.93% to 90,03%. This result is not too surprising since the 50% assistance produced already significant results. It is remarkable that such a system helps to reach over 90% of annotation correctness on average.

Hypothesis Bi "A 10% correct assistance system decreases the average time needed for one annotation compared to annotations done without the assistance."

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system did (statistically) not support the annotation task in terms of tempo, when it made only 10% correct (and 90% wrong) suggestions. The mean difference between blocks with assistance and blocks without is 0.51 seconds. Since this value is greater than zero, the assistance system with 10% correct suggestions did (on a descriptive level) cause a longer time needed to make a single correct annotation. The baseline there is 8.19 seconds without an assistance.

Hypothesis Bii "A 50% correct assistance system decreases the average time needed for one annotation compared to annotations done without the assistance."

$\Rightarrow p < \alpha$; this result **is significant**; the $H0$ is rejected. Consequently the assistance system did (statistically) support the annotation task in terms of tempo, when it made 50% correct (and 50% wrong) suggestions. On average, the annotations were done about 1.7s (per annotation) faster with this assistance than without it. This is a reduction of the baseline from 8.19 seconds to 6,49 seconds per correct annotation.

Hypothesis Biii "A 90% correct assistance system decreases the average time needed for one annotation compared to annotations done without the assistance."

$\Rightarrow p < \alpha$; this result **is significant**; the $H0$ is rejected. Consequently the assistance system did (statistically) support the annotation task in terms of tempo, when it made 90% correct (and 10% wrong) suggestions. On average, the annotations were done about 1.9s (per annotation) faster when this assistance was present than without it. This is a reduction of the baseline from 8.19 seconds to 6,29 seconds per correct annotation, or a decrease of more than 23% of the time.

Hypothesis Ci "A 10% correct assistance system decreases the miss rate per block compared to annotating a block of text done without the assistance."

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system did (statistically) not support the annotation task in terms of

reducing the rate of missed annotations, when it made only 10% correct (and 90% wrong) suggestions. The mean difference between blocks with assistance and blocks without is -1.55%. Descriptively this is a step in the right direction, but since it is not significant, such an assistance would not make an impact. The baseline of the miss rate is 7.69% of missed annotations without an assistance.

Hypothesis Cii "A 50% correct assistance system decreases the miss rate per block compared to annotating a block of text done without the assistance."

$\Rightarrow p < \alpha$; this result **is significant**; the $H0$ is rejected. Consequently the assistance system did (statistically) support the annotation task in terms of reducing the rate of missed annotations, when it made 50% correct (and 50% wrong) suggestions. On average, the rate of missed annotations is decreased of about 2.8% if the assistance was present. This is a reduction of the baseline from 7.69% to 4,89% of overall missed annotations. This is an improvement of more than 36%.

Hypothesis Ciii "A 90% correct assistance system decreases the miss rate per block compared to annotating a block of text done without the assistance."

$\Rightarrow p < \alpha$; this result **is significant**; the $H0$ is rejected. Consequently the assistance system did (statistically) support the annotation task in terms of reducing the rate of missed annotations, when it made 90% correct (and 10% wrong) suggestions. On average, the rate of missed annotations is decreased of about 3.94% if the assistance was present. This is a reduction of the baseline from 7.69% to 3,75% of overall missed annotations. This is an improvement of more than 51%.

Hypothesis Di "The 50% correct assistance system will increase the influence it has on correctness compared to the 10% correct assistance system."

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system in the levels 10% and 50% are, statistically, not different regarding their influence on the correctness of the participants' annotations.

Hypothesis Dii "The 90% correct assistance system will increase the influence it has on correctness compared to the 50% correct assistance system."

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system in the levels 50% and 90% are, statistically, not different regarding their influence on the correctness of the participants' annotations.

Hypothesis Ei "The 50% correct assistance system will increase the influence it has on tempo compared to the 10% correct assistance system."

Table 11: List of hypotheses, highlighting significant results.

	correctness	tempo	misses
10% correct Assistance	increase ✗	decrease ✗	decrease ✗
50% correct Assistance	increase ✓	decrease ✓	decrease ✓
90% correct Assistance	increase ✓	decrease ✓	decrease ✓
10% vs. 50%	increase ✗	decrease ✗	decrease ✗
50% vs. 90%	increase ✗	decrease ✗	decrease ✗

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system in the levels 10% and 50% are, statistically, not different regarding their influence on the tempo of the participants' annotations.

Hypothesis Eii "The 90% correct assistance system will increase the influence it has on tempo compared to the 50% correct assistance system."

Since $p > \alpha$, this result is **not significant**; the $H0$ is not rejected. Consequently the assistance system in the levels 50% and 90% are, statistically, not different regarding their influence on the tempo of the participants' annotations.

Hypothesis Fi and Fii "The 50% correct assistance system will increase the influence it has on the miss rate compared to the 10% correct assistance system."

Using the ANOVA we did not find a significant main effect of the level of the assistance system. Hence we did no further investigation and did not compute contrasts between the the levels of the assistance. The Fi related $H0$ as well as the Fii related $H0$ are not rejected.

Interestingly, we found a significant main effect of the block, expressing the point of time of measurement (the first half or the second half). This shows that the participants varied in their success in identifying all annotations in the first and in the second half of the study. A learning effect or a tiredness effect could explain this observation. Since we had no hypothesis regarding these effects, we did not analyze this any further.

Summary To summarize these results, Table 11 shows which of the hypotheses were significant. Compare Table 4.

5.1.1. Questionnaire Evaluation: Perceived Workload

This analysis is purely explorative. We established no hypotheses regarding the questionnaire.

assistance level 10% Since $p > \alpha$, this result is **not significant**. Consequently, the assistance system did not (statistically) support the annotation task in terms of reducing the perceived workload of the participants, when it made only 10% correct (and 90% wrong) suggestions. This is not surprising when we put this in relation to the results we obtained from our hypotheses tests regarding the 10% assistance. None of these tests were significant and we concluded that such a system does not support the task of annotation. Therefore it seems consistent that this system does not significantly influence the perceived workload of the participants.

assistance level 50% Since $p > \alpha$, this result is **not significant**.
Consequently the assistance system did not (statistically) support the annotation task in terms of reducing the perceived workload of the participants, when it made only 50% correct (and 50% wrong) suggestions. We would expect a significant impact on the perceived workload based on what we have analyzed using the same system so far – this is the first test with the 50% correct assistance that is not statistically significant. Thus the influence this system has is with an average of -0.32 points at a 7 point Likert scale still very marginal.

assistance level 90% $\Rightarrow p < \alpha = 0.05$; this result **is significant**.
Consequently, the assistance system did (statistically) support the annotation task in terms of reducing the perceived workload of the participants, when it made 90% correct (and 10% wrong) suggestions. On average, the perceived workload of the participants decreased about 0.7 points (at a 7 point Likert scale) when this assistance was present. The assistance system with 90% correct annotations does influence the perceived workload significantly. This seems plausible as the task of overseeing very correct pre-annotations is not very demanding but rather tedious.

ANOVA The results of the ANOVA testify a significant main effect of the level of the assistance. As we tested beforehand using the t-tests, we found, that the 90% correct assistance has a significant impact on the perceived workload of the participants. Besides this, there is no significant main effect of the block (the point of time of measurement) and no significant interaction effect of the level and the block.

5.1.2. Questionnaire Evaluation: Perceived Monotony

assistance level 10% Since $p > \alpha$, this result is **not significant**.
Consequently, the assistance system did not (statistically) support the annotation task in terms of reducing the perceived monotony of the participants when it made only 10% correct (and 90% wrong) suggestions. Similar to the result of the 10% assistance regarding the perceived workload, this is not surprising. The suggestions of this assistance system have too little impact to assist the annotation task. On a descriptive level, the average difference between blocks with and without assistance is positive and therefore impairs the perceived monotony.

assistance level 50% Since $p > \alpha$, this result is **not significant**.
Consequently the assistance system did (statistically) not support the annotation task in terms of reducing the perceived monotony of the participants when it made only 50% correct (and 50% wrong) suggestions. It is notable that, descriptively, the average difference between blocks with and without assistance is reduced by about 0.34 points of the Likert scale. Unfortunately this is not a statistically significant result.

assistance level 90% Since $p > \alpha$, this result is **not significant**.
Consequently, the assistance system did not (statistically) support the annotation task in terms of reducing the perceived monotony of the participants when it made 90% correct (and 10% wrong) suggestions. Similar to the results of the 10% assistance, this system seems – on a descriptive level – to impair the perceived monotony. A very bad and a very good assistance seem to bore the participant. One because it is frustratingly incorrect, the other because the assistance seems to not leave anything to annotate over.

Because none of the t-tests showed any significant results we did not investigate the data regarding the perceived monotony any further. We conclude, that all the different levels of the system did not influence the participants in this dimension; the task of annotating text seems to be truly monotonous.

5.1.3. Questionnaire Evaluation: Perceived Reliability of the Assistance

$\Rightarrow$ The p-value of the level of the assistance system is less than our significance level of $\alpha = 0.05$, this result **is significant**; we found a main effect of the level of the assistance. Besides this there is no significant main effect of the block (the point of time of measurement) and no significant interaction effect of the level and the block. No main effect of the block means that the perceived reliability of the assistance did not change over time. To be exact: It did not change between the first and the second halves. The main effect of the level of the assistance is plausible too – the better the assistance (regarding its level), the more reliable the participants perceived it.

On the 7 point Liker scale, the average score of the 10% assistance was 2.68; the 50% assistance was rated with 3.89 on average and the 90% assistance was rated with a score of 4.93 on average. On the one hand, the 10% assistance was rated quiet fair if we take into account that it had a mistake rate of 90%. On the other hand, the 90% correct assistance was rated disproportionately low, given the fact that it made 90% correct suggestions.

5.1.4. Questionnaire Evaluation: Perceived Correctness of the Assistance

$\Rightarrow$ The p-value of the level of the assistance system is less than our significance level of $\alpha = 0.05$, this result **is significant**; we found a main effect of the level of the assistance. Besides this, there is no significant main effect of the block (the point of time of measurement) and no significant interaction effect of the level and the block. These results are comparable to the evaluation of the perceived reliability: The perceived correctness did not change over time (no main effect of the block) and its correctness was rated proportionally to its actual correctness (main effect of the level).

It is remarkable that the participants again overestimated the poorest assistance but underestimated the best one. They did rate the 50% and the 90% correct assistance quite well (they missed their actual correctness by 7.32% and 10,25%), but overestimated the 10% correct assistance by almost 26%. We assume that the 90% mistakes that the poorest assistance made were not perceived as bad as they actually are. Since it made different mistakes (see Section 2.3), we believe that they are not equally hard to improve.[27] We assume for example that a correct labeled chunk with just a little offset in its span is fairly easy to improve – for example easier than a missed annotation. Since 17.55% (of the 90% mistakes that the poorest assistance system made, see Section 2.3) are annotations with just a wrong span, it seems plausible that such mistakes were not perceived to be as wrong as they actually are.

5.2. Method Discussion

We kept a constant order in which the texts were presented. All the participants saw the same texts in the same order. This decision was made since it was hard to separate the corpus into blocks and documents, equal in count of annotations for all the participants. The paragraphs of the 14 different texts were displayed in the correct, coherent order to maintain a logical connection from one paragraph to the next. To keep things simple we fixed an order of paragraphs applicable to all participants. As a consequence we were not able to show for certain that texts in all blocks are equally demanding in terms of cognitive load. This means that if there was a difference in the perceived complexity of the blocks, it could have influenced the findings we interpreted as caused

[27] In fact, we encourage the interested reader to analyze our data set if the different mistakes are differently hard to find and to improve. See Section 5.4.

by the assistance system. Since we shuffled the presence of the assistance system, we minimized this impact; but in order to replicate and improve our findings we suggest to find a way to shuffle the order of the paragraphs as well.

5.3. Practical Implications

An assistance system achieving 90% of correctness is technically not realistic. The best NER systems for the German language currently reach around 80% in precision and recall [24]. If we could create an assistance with an correctness of 90%, we would probably already have a sufficient training data set. However, to craft an assistance system with 50% correctness is definitely feasible. We showed that such a system already helps to increase the performance regarding our three performance dimensions.

The 50% assistance leads to about 3.7% percentage points more correct annotations. The baseline here is especially interesting: Without any assistance, our participants achieved 83.9% correctness on average. Using the 50% accurate assistance this average correctness increases to around 87,6%. Reaching the last percentages is always harder than improving the early percentages – this systems helps getting closer to a fully correct annotated data set (although the system itself made far less correct suggestions).[28]

We would not only improve the quality of the annotations but also save time *(= costs)* with the proposed assistance: A typical corpus for NLP tasks contains tens of thousands of sentences and annotations. The GermEval 2014 NER corpus [35] for example is made up of 31.297 sentences and 37.926 named entity annotations. Our participants spent 8.2 seconds on average per correct named entity annotation – without any assistance. If they had annotated the GermEval corpus at this pace, they would have spent more than two weeks of full time work on this task.[29] We showed that we can reduce the average time per annotation by about 1.7 seconds if we if we employ an assistance system with only 50% correctness. This saves more than two work days in the illustrated example, a little more than 20%.[30]

Finally the same assistance system will reduce the average rate of missed annotations notably about 36%.

Should the Assistance Always be Present? The lowest of the tested assistance levels, providing 10% accurate suggestions, did not show any significant impact. Looking at the graphs of our data on a descriptive level, a very poor assistance does not help but impair the annotator's performance. Thus we recommend to use the assistance system only if its correctness can be assured to be around 50% or more. Only like this our

[28] This data describes the average performance of a single annotator. In a real world example we would be able to increase this numbers by applying an *inter annotator agreement* as, for example, described in [45].

[29] $(8.2\ seconds \times 37.926\ annotations)/3.600\ seconds\ per\ hour/8\ hours\ per\ day = 10.8\ days$

[30] $(6.5\ seconds \times 37.926\ annotations)/3.600\ seconds\ per\ hour/8\ hours\ per\ day = 8.6\ days$

findings support the assumption of improving performance by employing an assistance as described.

5.4. Further Research Questions

During the analysis of our data we encountered two further research questions:

A Linear Correlation of the Assistance's Level and its Impact? We chose a rather simple setup with three distinct correctness levels to create a reliable first impression of the impact such an assistance has on the performance of annotators. These ranged from poor to pessimistically realistic to unrealistic in their quality. We showed that the 50% accurate assistance provides significant improvements – thus the assistance should only be enabled when its correctness can be ensured.

Our findings result in the idea that there might be a linear correlation of the correctness of the assistance and the impact it has on human annotators. This thesis should be validated in a further study. Therefore we suggest to investigate a linear correlation, to find the average interception of the impact and the annotator's performance and to validate this crucial "break even point" with accordingly adjusted levels of the assistance. At this point it's safe to employ the assistance system; and with a growing number of annotations the system will be able to improve itself and thus create more accurate suggestions. In other words: Once we find out the according intercept we can probably benefit from an assistance before it reaches the 50% correctness label.

Difference of Mistakes? Antother interesting task is the analysis of the different error classes that can occur while annotating and by creating automated suggestions.[31] We assume they are not equally easy to correct for annotators. This first question can be answered by a further investigation of the data we collected. Next, we hope that it is possible to derive special annotation instructions to raise awareness of the most difficult error classes. This should lead to a further improvement of correctness of annotations while using an assistance as we proposed.

5.5. Conclusion

Our main hypotheses (groups A, B and C) are supported by the data we analyzed. The sole exception are the hypotheses regarding the assistance system with an correctness of 10%. A metaphor to describe this situation would be a spell checker software, set to the wrong language. It does underline words as it should, but most of the suggestions are wrong. This also applies to the assistance on its poorest level. It is reasonable that such a system does not support the task – according to our findings we can say that a

[31] See Table 2 for an overview of possible annotation mistakes.

minimum level of 50% correctness is needed to do so. What we did not expect was not being able to find a significant improvement when comparing the different system levels with each other. We only examined the relation between the 10% and the 50% correct system, as well as the 50% and the 90% correct system.

Our findings can directly be applied to the annotation systems that are already available – if they can be extended with an assistance system as described. Especially the DALPHI framework profits from our findings – as all components needed to use the system we described in this work are already present.

The study supports the findings of Day et al. [9], especially their assertion on productivity improvement of pre-annotations. Our results extend the findings of Day et al. [9] even further with the analysis of the three performance dimensions we identified: Correctness, tempo and miss rate.
We also concur with Settles [10] in the aspect of reconstructing the benefits AL provides for the machine as well as for the human annotator. Furthermore, we refined his conclusion by providing evidence for a pre-labeling system improving the human's performance in three different dimensions that were mentioned above.

This work successfully showed possibilities to increase the efficiency of text based training data generation. We hope that this study, the DALPHI framework and the code we provided, will help to create a larger amount of diverse training data sets. Various languages for example are currently underrepresented in terms of available training corpora. We hope that this work will help to bring (supervised) ML technologies to more languages and therefore to more people.

References

[1] H. Cunningham, D. Maynard, and K. Bontcheva, *Text processing with gate*. Gateway Press CA, 2011.

[2] J. Carletta, S. Evert, U. Heid, J. Kilgour, J. Robertson, and H. Voormann, "The NITE XML toolkit: flexible annotation for multimodal language data", *Behavior Research Methods*, vol. 35, no. 3, pp. 353–363, 2003.

[3] T. Morton and J. LaCivita, "WordFreak: an open tool for linguistic annotation", in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations-Volume 4*, Association for Computational Linguistics, 2003, pp. 17–18.

[4] C. Biemann, K. Bontcheva, R. E. de Castilho, I. Gurevych, and S. M. Yimam, "Collaborative Web-based Tools for Multi-layer Text Annotation", *The Handbook of Linguistic Annotation', Text, Speech, and Technology book series, Springer Netherlands*, 2017.

[5] K. Bontcheva, H. Cunningham, I. Roberts, A. Roberts, V. Tablan, N. Aswani, and G. Gorrell, "GATE Teamware: a web-based, collaborative text annotation framework", *Language Resources and Evaluation*, vol. 47, no. 4, pp. 1007–1029, 2013. [Online]. Available: `http://link.springer.com/article/10.1007/s10579-013-9215-6`.

[6] S. M. Yimam, I. Gurevych, R. E. de Castilho, and C. Biemann, "WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations.", in *ACL (Conference System Demonstrations)*, 2013, pp. 1–6. [Online]. Available: `http://www.aclweb.org/website/old_anthology/P/P13/P13-4.pdf#page=13`.

[7] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "BRAT: a web-based tool for NLP-assisted text annotation", in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2012, pp. 102–107.

[8] S. Ananiadou and J. Tsujii, "stav: text annotation visualiser", 2012. [Online]. Available: `http://www.anlp.jp/proceedings/annual_meeting/2012/pdf_dir/P1-38.pdf`.

[9] D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain, "Mixed-initiative development of language processing systems", in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, 1997, pp. 348–355.

[10] B. Settles, "Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances", in *Proceedings of the conference on empirical methods in natural language processing*, Association for Computational Linguistics, 2011, pp. 1467–1478.

[11] P. F. Drucker, “Managing for Business Effectiveness”, *Harvard Business Review*, pp. 53–60, 1963. [Online]. Available: `https://hbr.org/1963/05/managing-for-business-effectiveness`.

[12] W. O. Galitz, *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons, 2007.

[13] R. Agerri and G. Rigau, “Robust multilingual Named Entity Recognition with shallow semi-supervised features”, *Artificial Intelligence*, vol. 238, pp. 63–82, 2016.

[14] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification”, *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[15] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition”, in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, 2003, pp. 142–147.

[16] B. Settles, “Active learning literature survey”, *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.

[17] F. Olsson, “A literature survey of active machine learning in the context of natural language processing”, 2009.

[18] M. Bächle and P. Kirchberg, “Ruby on rails”, *IEEE software*, vol. 24, no. 6, 2007.

[19] “Web Services Architecture: Relationship to the World Wide Web and REST Architectures”, *W3C Working Group Note 11 February 2004*, 2004. [Online]. Available: `https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest`.

[20] T. Bray, “The javascript object notation (json) data interchange format”, 2014.

[21] E. L. Bird Steven and E. Klein, “Natural Language Processing with Python: Extracting Information from Text”, 2009. [Online]. Available: `http://www.nltk.org/book/ch07.html`.

[22] P. Nolte, “Natural Language Toolkit: Classifier Based German Tagger”, 2011. [Online]. Available: `https://github.com/ptnplanet/NLTK-Contributions/tree/master/ClassifierBasedGermanTagger`.

[23] J. Schreiber, “A German word list for GNU Aspell”, 2017. [Online]. Available: `https://sourceforge.net/projects/germandict/files/`.

[24] D. Benikova, C. Biemann, M. Kisselew, and S. Pado, “Germeval 2014 named entity recognition: Companion paper”, *Proceedings of the KONVENS GermEval Shared Task on Named Entity Recognition, Hildesheim, Germany*, pp. 104–112, 2014.

[25] E. T. Jaynes, “Information theory and statistical mechanics”, *Physical review*, vol. 106, no. 4, pp. 620–630, 1957.

[26] K. Nigam, J. Lafferty, and A. McCallum, “Using maximum entropy for text classification”, in *IJCAI-99 workshop on machine learning for information filtering*, vol. 1, 1999, pp. 61–67.

[27] H. L. Chieu and H. T. Ng, "Named Entity Recognition: A Maximum Entropy Approach Using Global Information", in *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, ser. COLING '02, Taipei, Taiwan: Association for Computational Linguistics, 2002, p. 2. DOI: 10.3115/1072228.1072253. [Online]. Available: http://dx.doi.org/10.3115/1072228.1072253.

[28] H. Daumé III, "MegaM: Maximum entropy model optimization package", *ACL Data and Code Repository, ADCR2008C003*, vol. 50, 2008.

[29] H. Daumé III, "Notes on CG and LM-BFGS optimization of logistic regression", Aug. 2004.

[30] NLTK Project, "NLTK 3.2.4 documentation: nltk.chunk.named_entity", 2017. [Online]. Available: http://www.nltk.org/_modules/nltk/chunk/named_entity.html.

[31] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization", *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[32] J. R. Shewchuk *et al.*, *An introduction to the conjugate gradient method without the agonizing pain*, 1994.

[33] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification", *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[34] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation", 2011.

[35] "GermEval 2014 Named Entity Recognition Shared Task", 2014. [Online]. Available: https://sites.google.com/site/germeval2014ner/data.

[36] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition", *arXiv preprint arXiv:1603.01360*, 2016.

[37] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, "G* Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences", *Behavior research methods*, vol. 39, no. 2, pp. 175–191, 2007.

[38] J. Cohen, *Statistical power analysis for the behavioral sciences (revised ed.)* 1977.

[39] Z. F. Andy Field Jeremy Miles, *Discovering statistics using R*. Sage Publications, 2012.

[40] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)", *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[41] H. Levene *et al.*, "Robust tests for equality of variances", *Contributions to probability and statistics*, vol. 1, pp. 278–292, 1960.

[42] P. Mair and R. Wilcox, *Robust Statistical Methods in R Using the WRS2 Package*, 2016.

[43] B. Welch, “On the comparison of several mean values: an alternative approach”, *Biometrika*, vol. 38, no. 3/4, pp. 330–336, 1951.

[44] C. E. Bonferroni, *Teoria statistica delle classi e calcolo delle probabilita*. Libreria internazionale Seeber, 1936.

[45] T. Brants, “Inter-annotator Agreement for a German Newspaper Corpus.”, in *LREC*, 2000.

[46] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, “A Maximum Entropy Approach to Natural Language Processing”, *Comput. Linguist.*, vol. 22, no. 1, pp. 39–71, Mar. 1996, ISSN: 0891-2017. [Online]. Available: `http://dl.acm.org/citation.cfm?id=234285.234289`.

6. Appendix

A. Analysis Scripts and the Data Set

The data set we created from the annotated texts of the participants is available in the CSV and in the JSON format. It can be downloaded here (last date of retrieval for the four following links: August 4, 2017):
`https://github.com/RGreinacher/bachelor-thesis/tree/master/Studie/Data`

The R scripts we wrote to analyze the data set and to create plots can be found here:
`https://github.com/RGreinacher/bachelor-thesis/tree/master/Apps/R`

The assistance system simulation we used for the study is available as well. This software can be started and registered in a DALPHI instance as an iteration- and merge service:
`https://github.com/RGreinacher/bachelor-thesis/tree/master/Apps/Dalphi%20Service`

The text we used for the study (see Section C), including the pre-annotations for all the study participants are available here:
`https://github.com/RGreinacher/bachelor-thesis/tree/master/Korpora/VP%20vorbereitet`.

B. The Development of the DALPHI framework

DALPHI was developed by the Berlin based startup Implisense GmbH. The author of this work is one of the developers of the software project. The other develpoer, a fellow student at *Technische Universität Berlin*, is Arik Grahl. Both developers worked at the company in part time to realize the project. The exact amount of contributions, as well as the detailed share of development contributed by the author of this work can be looked up via the following links to the public *GitHub* project pages:

1. `https://github.com/Dalphi` (last date of retrieval: August 11, 2017)
2. `https://github.com/Dalphi/dalphi/commits?author=RGreinacher` (last date of retrieval: August 11, 2017)
3. `https://github.com/Dalphi/interface-paragraph_classification/commits?author=RGreinacher` (last date of retrieval: August 11, 2017)
4. `https://github.com/Dalphi/interface-questionnaire/commits?author=RGreinacher` (last date of retrieval: August 11, 2017)
5. `https://github.com/Dalphi/service-ner_iterate/commits?author=RGreinacher` (last date of retrieval: August 11, 2017)

6. https://github.com/Dalphi/interface-ner_complete/commits?author=RGreinacher (last date of retrieval: August 11, 2017)

C. The Study Texts

All of the following websites were last retrieved on July 31, 2017.

1. "Aperto erweitert die Führungsebene" (17.05.2013 - 17:09) https://www.pressebox.de/pressemitteilung/aperto-ag/Aperto-erweitert-die-Fuehrungsebene/boxid/597473
2. "Ausmisten 3.0 dank neuer Smartphone App: Mit dem „reBuy.de Scanner" im Turbogang gebraucht Medienartikel zu Geld machen" (01.10.2013 - 10:00), http://presse.rebuy.de/2013/09/30/ausmisten-3-0-dank-neuer-smartphone-app-mit-dem-rebuy-de-scanner-im-turbogang-gebrauchte-medienartikel-zu-geld-machen/
3. "https://wice.de/crm-zertifizierung-wice-bereits-zum-dritten-mal-mit-top-platzierung-ausgezeichnet/6202/" (12.09.2011 - 17:16) https://wice.de/crm-zertifizierung-wice-bereits-zum-dritten-mal-mit-top-platzierung-ausgezeichnet/6202/
4. "CytoTools AG: CytoTools AG und DermaTools Biotech GmbH mit erfolgreichen Kapitalmaßnahmen i schwierigem Marktumfeld" (22.12.2010 - 17:23) http://www.dgap.de/dgap/News/corporate/cytotools-cytotools-und-dermatools-biotech-gmbh-mit-erfolgreichen-kapitalmassnahmen-schwierigem-marktumfeld/?newsID=654068
5. "tolingo.de: Schüler lassen Latein im Internet übersetzen" (21.11.2008 - 12:57) https://www.pressebox.de/inaktiv/tolingo-gmbh/tolingode-Schueler-lassen-Latein-im-Internet-uebersetzen/boxid/219892
6. "Gabriel verkündet Durchbruch - So soll Kaiser's Tengelmann doch noch gerettet werden" (01.11.2016 - 10:35) http://www.spiegel.de/wirtschaft/unternehmen/kaiser-s-tengelmann-edeka-uebernimmt-die-kette-rewe-einige-filialen-in-berlin-a-1119064.html
7. "Gemeinsam shoppen mit Schauspielerin Jenny Winkler" (25.08.2008 - 12:28) http://www.regenta-verlag.de/stadtmagazin/stadt/henstedtulzburg/?c=8&subaction=showfull&id=1220872464&ucat=2
8. "Hauptversammlung der SolarWorld AG beschließt Dividende von 0,19 Euro" (24.05.2011 - 18:37) http://www.solarworld.de/konzern/investor-relations/news-amp-veroeffentlichungen/corporate-news/single-ansicht/article/hauptversammlung-der-solarworld-ag-beschliesst-dividende-von-eur-019/

9. “Mindjet-Aktion für Bildung und Forschung: MindManager 8 für nur 29 Euro” (02.06.2010 - 11:47)
https://www.mindjet.com/de/press-release/02-06-2010-mindjet-aktion-fur-bildung-und-forschung-mindmanager-8-fur-nur-29-euro/

10. “Online-Debüt für scorio Pro, den Premium-Account von scorio” (08.11.2011 - 11:42)
http://www.scorio.com/web/scorio/press_201203

11. “meinestadt.de und eBay Kleinanzeigen kooperieren” (29.06.2010 - 08:57)
https://www.pressebox.de/pressemitteilung/allesklarcom-ag/meinestadtde-und-eBay-Kleinanzeigen-kooperieren/boxid/356017

12. “CHECK24-Energiesparratgeber mindern CO2-Emission um 881 Tonnen” (26.01.2012 - 08:40)
http://www.check24.de/files/p/2012/5/7/2/173_2012-01-26_check24_pm_co2online_portalpartner_des_jahres.pdf

13. “naturstrom setzt auf saubere E-Mobilität” (06.07.2011 - 17:45)
https://www.naturstrom.de/ueber-uns/presse/news-detail/naturstrom-setzt-auf-saubere-e-mobilitaet/

14. “Leseberg macht Volkswagen fit für den Winter” (19.11.2009 - 14:29)
http://www.openpr.de/news/372738/Leseberg-macht-Volkswagen-fit-fuer-den-Winter.html

The complete collection of text files the study participants annotated can be found here: https://github.com/RGreinacher/bachelor-thesis/tree/master/Korpora/source/raw (last date of retrieval: August 11, 2017)

D. The Questionnaires

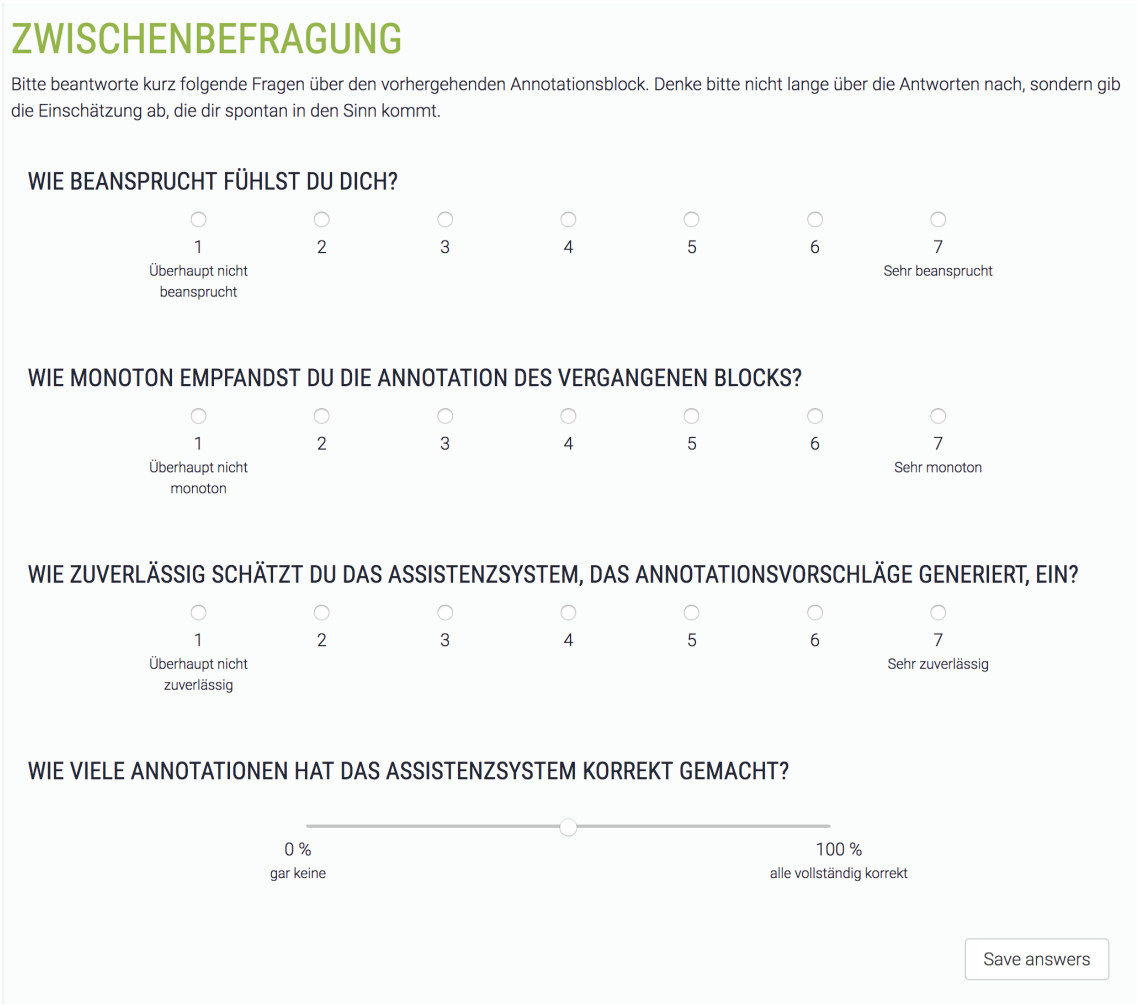


Figure 21: Intermediate questionnaire with four items, presented after blocks with assistance.

Figure 22: Demographic questionnaire that was asked after the fourth block of trials.

E. Study Material

E.1. Declaration of Consent

Herzlich willkommen zur Studie

Vielen Dank, dass du dir Zeit für diesen Versuch nimmst. Solltest du Fragen haben, kannst du diese jederzeit gerne stellen. Die Teilnahme ist freiwillig und du kannst sie jederzeit ohne Angabe von Gründen beenden.

In dem nachfolgenden Versuch sollen Namen in Texten identifiziert und annotiert (= den Text mit Markierungen für Personen- und Organisationsnamen versehen) werden. Der Versuch ist in vier Blöcke unterteilt. In zweien wird ein Assistenzsystem bereits Markierungen in den Text eingefügt haben. Dies sind Vorschläge zu den Namen, die erkannt werden sollen und können fehlerhaft sein. Deine Aufgabe ist es, alle Dokumente korrekt zu annotieren. Falls Fehler auftreten, bitten wir dich diese zu korrigieren. Ziel ist es herauszufinden, ob das Assistenzsystem dich bei der Arbeit der Annotation unterstützt.

Deine Antworten werden ausschließlich in anonymisierter Form gespeichert und weiterverarbeitet. Die Anonymisierung wird durch die Zuweisung eines Versuchspersonencodes zu den erfassten Datensätzen sichergestellt. Ferner ist nur eine zusammengefasste Auswertung der erhobenen Daten geplant. Persönliche Daten wie Name, Telefonnummer oder Anschrift werden nicht erfasst. Solltest du dazu Fragen haben, kannst du dich jederzeit an uns wenden. Die Kontaktdaten findest du am Ende dieses Blattes.

Damit du an den Versuchen teilnehmen kannst, möchten wir dich bitten, dir die folgende Freigabe aufmerksam durchzulesen und diese zu unterschreiben. Solltest du im Nachhinein Bedenken haben, so kannst du diese Freigabe jederzeit widerrufen. Deine Daten werden dann gelöscht (notiere dir hierzu deinen Versuchspersonencode).

- Ich wurde über den Zweck der Studie informiert und habe eine Möglichkeit jederzeit weitere Informationen einzuholen.
- Meine Versuchsteilnahme ist freiwillig.
- Mir ist bewusst, dass ich die Teilnahme an dem Versuch jederzeit und ohne Angabe von Gründen abbrechen kann.
- Mir ist bewusst, dass ich jederzeit eine Löschung der erhobenen Daten verlangen kann.

Berlin, den

_____ _____

Datum Unterschrift VersuchsteilnehmerIn

Robert Greinacher
Hagelberger Straße 15
10965 Berlin
studie@robert-greinacher.de
0173 / 310 74 52

Anna Trapp
Kognitionspsychologie und Kognitive Ergonomie - TU Berlin
Sekr. MAR 3 -2
Marchstr. 23
10587 Berlin
anna.k.trapp@tu-berlin.de
030 / 31 42 52 79

E.2. Annotation Guidelines

Annotationsrichtlinien

Lies dir dieses Dokument für einen Überblick vor Beginn der Annotationsaufgabe aufmerksam durch. Anschließend dient es als Hilfestellung während der Annotation. Wenn die Studie beginnt, können keine Fragen der Art „welches Label ist korrekt?“ beantwortet werden.
Es sollen Personennamen (in grün) und Firmen-, bzw. Organisationsnamen (in blau) im Text markiert werden. Dafür stehen zwei Farben zur Verfügung.

Personennamen:

Grundsätzlich sind Personennamen alle Vorkommen von Namen für Menschen. Ausnahmen sind Namen für Organisationen oder Firmen, die nach einem Menschen benannt wurden. Es gilt hier zu unterscheiden was jeweils gemeint ist: Geht es um eine Person, so ist der Name als Personenname zu kennzeichnen.
Es werden lediglich die Namen selbst, keine Titel, Anreden oder andere Beisätze annotiert.

Beispiele:
- „Sonja van der Linden wohnt in Berlin.“
- „In der Gärtnerei hat Herr Dr. Achim viel Arbeit.“
- „Vielen Dank für Ihre Bewerbung, sagte die Geschäftsführerin Gabriele Kohler.“

Manchmal werden Namen in anderen grammatikalischen Formen verwendet. Diese bezeichnen dennoch eine Person und werden demnach als Personenname annotiert:

- „Darauf antwortete Merkels Kanzleramt prompt.“
- „Petras Kabeltechnik hat gute Preise.“ (Petra arbeitet bei der Kabeltechnik AG)

Firmen- bzw. Organisationsnamen:

Diese Namen bezeichnen Firmen- und Organisationsnamen ohne Artikel, jedoch mit Rechtsform

- **Firmen**
  - Sie arbeitet bei der Deutschen Bahn AG.
  - Sonjas Café am Marktplatz verkauft leckeres Eis.
- **Firmen mit Produkten** werden getrennt markiert
  - Sie liebt ihre Mercedes A-Klasse.
  - Er chattet oft mit dem Facebook Messenger.
- **Organisationen**
  - Er dankte dem Roten Kreuz nach dem Unfall.
  - Sie besucht das Gutenberg Gymnasium.
  - Der Deutsche Bundestag lädt zum Tag der offenen Tür.
  - Der Außenminister (SPD) lädt zu Kaffee und Kuchen.

Ist eine Einrichtung nach einem Menschen benannt, so wird dieser Name als Firma markiert (bsp: die Claudia Kleber GmbH). **Somit hat der Firmenname Vorrang vor dem Personennamen.**
Ausnahmen sind Namen, die zwar auch einer Firma einen Namen geben, an dieser Stelle aber die Person selbst meinen (bsp: Claudia Kleber arbeitet bei der Claudia Kleber GmbH).
Rechtsformen (AG, GmbH, GbR) gehören auch zum Firmenname und werden (falls angegeben) mit markiert.

Organisationen werden nur markiert, wenn sie durch die Nennung konkret identifiziert werden können:
Beispiel: „...in Zusammenarbeit mit den jeweiligen Berufsgenossenschaften.“
Hierbei sind die Berufsgenossenschaften keine Organisation, da sie unspezifisch verwendet werden.

Wichtig ist der Satzzusammenhang:

Grundsätzlich werden alle Markierungen entsprechend ihrer Verwendung im Satzzusammenhang vorgenommen:

Beispiel: „Bei Zalando.de arbeiten viele MitarbeiterInnen.“
Der eigentliche Firmenname lautet zwar „Zalando SE“ (SE = Societas Europaea), doch die Internetadresse wird im Satzzusammenhang anstelle des Firmennamens verwendet.

Mehrdeutigkeiten:

Vorsicht bei Mehrdeutigkeiten. Deren Bedeutung muss ggf. aus dem Kontext erschlossen werden:

Organisationsnennung mit Rechtschreibfehler im Satz:
"Er dankt dem roten Kreuz"
Satz ohne Organisationsnennung:
"Er dankt dem roten Kreuz an der Tür jeden Tag"

Kompletter Firmenname:
"Ich werde es Sonjas Café am Marktplatz nennen, sagte die Geschäftsführerin."
Personenname, dann ein Firmenname:
"Ich schlage Sonjas Café am Marktplatz vor, vielleicht ist Sonja selbst auch da. "
Personenname, dann ein Straßenname und kein Firmenname:
"Lass uns in Sonjas Café am Marktplatz treffen, direkt neben dem Rathaus."
Personenname, dann ein Straßenname, dann der Firmenname:
"Sonjas Café am Marktplatz Vivaldi hat guten Kuchen."

E.3. User interface Functions

Funktionen des Annotationsinterfaces

Nach Use-Case

Annotiere ein Wort

Klicke mit dem Mauszeiger auf ein Wort. Dieses wird dadurch farbig hinterlegt.

Annotiere mehrere Worte

Drücke mit dem Mauszeiger auf ein Wort, lasse die Maustaste nicht los, und ziehe die Maus über ein vorhergehendes oder nachfolgendes Wort. Alle Worte von Beginn bis Ende des Klicks werden nun zu einer Annotation zusammengefasst.

Wähle eine Annotation aus

- *Maus:* Klicke mit dem Mauszeiger auf eine Annotation.
- *Tastatur:* Drücke die `Tab` Taste bis die zu verändernde Annotation ausgewählt ist. So werden alle Annotationen der Reihe nach hervorgehoben.

Verändere die Länge einer bestehenden Annotation

- *Maus:* Benutze die kleinen weißen Anfasser; drücke mit dem Mauszeiger auf einen Anfasser, halte die Maustaste gedrückt, und ziehe die Maus über ein vorhergehendes oder nachfolgendes Wort.
- *Maus:* Drücke mit dem Mauszeiger auf ein Wort innerhalb einer Annotation, halte die Maustaste gedrückt, und ziehe den Mauszeiger bis zu dem Wort, das die Annotation auch noch umfassen soll.
- *Tastatur:* Benutze die `Pfeiltasten` (`<-` und `->`) um eine Annotation am linken oder am rechten Ende zu erweitern. Wenn du dabei `Shift` („Großschreibtaste“) gedrückt hältst kannst du dieselben Tasten verwenden um eine Annotation zu verkleinern.

(Annotationen können nicht aus weniger als aus einem Wort bestehen.)

Wähle eine Kategorie für eine bestehende Annotation

- *Maus:* Klicke auf die entsprechende Kategorie die als Beispiele über dem Text angezeigt werden.
- *Tastatur:* Drücke die Taste `1` um die ausgewählte Annotation als Person zu markieren, oder drücke die Taste `2` um die ausgewählte Annotation als Firma / Organisation zu markieren.

Entferne eine bestehende Annotation

Drücke die `Backspace` Taste (Pfeil nach links, oberhalb von `Enter`).

(Tipp: Eine Annotation zu entfernen und dann neu auszuwählen ist manchmal einfacher als eine bestehende zu verändern.)