View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Edinburgh Research Explorer

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Agile Corpus Annotation in Practice: An Overview of Manual and Automatic Annotation of CVs

Citation for published version:
Alex, B, Grover, C, Shen, R & Kabadjov, M 2010, Agile Corpus Annotation in Practice: An Overview of Manual and Automatic Annotation of CVs. in Proceedings of the Fourth Linguistic Annotation Workshop. Association for Computational Linguistics, Uppsala, Sweden, pp. 29-37.

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of the Fourth Linguistic Annotation Workshop

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 28. Apr. 2017

Agile Corpus Annotation in Practice: An Overview of Manual and Automatic Annotation of CVs

Bea Alex Claire Grover Rongzhou Shen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

Mijail Kabadjov
Joint Research Centre
European Commission
Via E. Fermi 2749, Ispra (VA), Italy

Contact: balex@staffmail.ed.ac.uk

Abstract

This paper describes work testing agile data annotation by moving away from the traditional, linear phases of corpus creation towards iterative ones and by recognizing the potential for sources of error occurring throughout the annotation process.

1 Introduction

Annotated data sets are an important resources for various research fields, including natural language processing (NLP) and text mining (TM). While the detection of annotation inconsistencies in different data sets has been investigated (e.g. Novák and Razímová, 2009) and their effect on NLP performance has been studied (e.g. Alex et al. 2006), very little work has been done on deriving better methods of annotation as a whole process in order to maximize both the quality and quantity of annotated data. This paper describes our annotation project in which we tested the relatively new approach of agile corpus annotation (Voormann and Gut, 2008) of moving away from the traditional, linear phases of corpus creation towards iterative ones and of recognizing the fact that sources of error can occur throughout the annotation process.

We explain agile annotation and discuss related work in Section 2. Section 3 describes the entire annotation process and all its aspects. We provide details on the data collection and preparation, the annotation tool, the annotators and the annotation phases. Section 4 describes the final annotation scheme and Section 5 presents inter-annotator-agreement (IAA) figures measured throughout the annotation. In Section 6, we summarize the performance of the machine-learning (ML)-based TM components which were trained and evaluated on the annotated data. We discuss our findings and conclude in Section 7.

2 Background and Related Work

The manual and automatic annotation work described in this paper was conducted as part of the TXV project. The technology used was based on TM components that were originally developed for the biomedical domain during its predecessor project (Alex et al., 2008b). In TXV we adapted the tools to the recruitment domain in a short time frame. The aim was to extract key information from curricula vitae (CVs) for matching applicants to job adverts and to each other. The TM output is visualized in a web application with search navigation that captures relationships between candidates, their skills and organizations etc. This web interface allows recruiters to find hidden information in large volumes of unstructured text.

Both projects were managed using agile, test-driven software development, i.e. solutions were created based on the principles of rapid-prototyping and iterative development cycles of deliverable versions of the TM system and the web application.[1] The same principles were also applied to other project work, including the manual annotation. The aim of this annotation was to produce annotated data for training ML-based TM technology as well as evaluating system components.

Collecting data, drawing up annotation guidelines and getting annotators to annotate this data in sequential steps is similar to the waterfall model in software engineering (Royce, 1970). This approach can be inefficient and costly if annotators unknowingly carried out work that could have been avoided and it can lead to difficulties if at the end of the process the requirements no longer match the annotations. Instead we applied agile software engineering methods to the process of creating annotated data. This is a relatively recent philosophy in software

[1] The agile software development principles are explained in the Agile Manifesto: http://agilemanifesto.org/

Proceedings of the Fourth Linguistic Annotation Workshop, ACL 2010, pages 29–37,
Uppsala, Sweden, 15-16 July 2010. ©2010 Association for Computational Linguistics

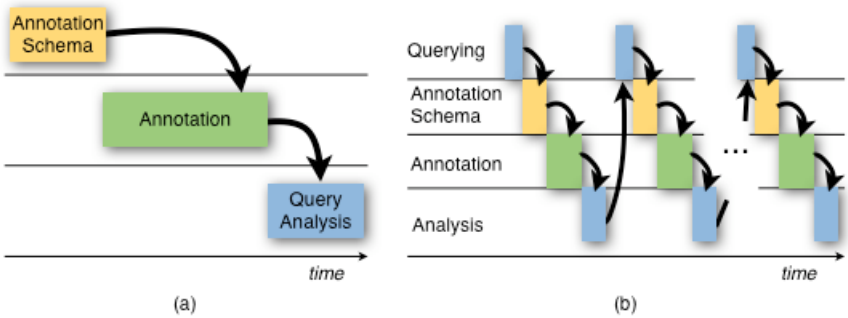

Figure 1: The phases of traditional corpus creation (a) and the cyclic approach in agile corpus creation (b). Reproduction of Figure 2 in Voormann and Gut (2008).

development which was inspired to overcome the drawbacks of the waterfall model. The idea of applying agile methods to corpus creation and annotation was first inspired by Voormann and Gut (2008) but was not tested empirically. Cyclic annotation was already proposed by Atkins et al. (1992) and Biber (1993) with a focus on data creation rather than data annotation. In this paper, we describe a way of testing this agile annotation in practice.

The idea behind an agile annotation process is to produce useable manually annotated data fast as well as discover and correct flaws in either the annotation guidelines or the annotation setup early on. Voormann and Gut (2008) propose query-driven annotation, a cyclic corpus creation and annotation process that begins with formulating a query. The main advantages of this approach are:

- The annotation scheme evolves over time which ensures that annotations are consistent and remain focussed on the research that is carried out. An iterative annotation process therefore improves the annotation guidelines but keeps the annotations suitable to the relevant research questions.
- Problems with the annotation guidelines, errors in the annotation and issues with the setup become apparent immediately and can be corrected early on. This can avoid difficulties later on and will save time and cost.
- Some annotation data is available early on.

Voormann and Gut compare the cyclical approach in agile annotation to traditional linear-phrase corpus creation depicted in Figure 1. In the following section we describe the annotation process in our project which followed the principles of agile corpus creation.

3 Annotation Process

This section provides an overview of all aspect involved in the annotation of a data set of CVs for various types of semantic information useful to recruiters when analysing CVs and placing candidates with particular jobs or organizations. We provide information on the data collection, the document preparation, the annotation tool and the annotation process following agile methods.

3.1 Data Collection

We automatically collected a set of CVs of software engineers and programmers which are publicly available online. This data set was created by firstly querying Google using the Google API[2] for word documents containing either the terms *"CV"*, *"resume"* or *"curriculum vitae"* as well as the terms *"developer"*, *"programmer"* or *"software"* but excluding documents containing the word *"template"* or *"sample"*. Furthermore, the query was restricted to a 3-month period from 30/03 to 30/06/2008.[3]

We automatically downloaded the Word documents returned by this query, resulting in a pool of 1,000 candidate CVs available for annotation. We split these documents randomly into a TRAIN, a DEVTEST and a TEST set in a ratio of approximately 64:16:20. We used the annotated TRAIN data for training ML-based models and deriving rules and the DEVTEST data for system development and optimization. We set aside the blind TEST set for evaluating the final performance of our named entity recognition (NER) and relation extraction (RE)

[2] http://code.google.com/apis/ajaxsearch

[3] The exact Google query is: '(CV OR resume OR "curriculum vitae") AND (developer OR programmer OR software) AND filetype:doc AND -template AND -sample AND daterange:2454466-2454647'.

CV data set				
Set	TRAIN	DEVTEST	TEST	ALL
Files	253	72	78	403
Annotations	279	84	91	454

Table 1: Number of files and annotated files in each section of the CV data set.

components (see Section 6).

The final manually annotated data set contains 403 files, of which 352 are singly and 51 doubly annotated, resulting in an overall total of 454 annotations (see Table 1). This does not include the files used during the pilot annotation. The doubly annotated CVs were used to determine inter-annotator agreement (IAA) in regular intervals (see Section 5).

Some of the documents in the pool were not genuine CVs but either job adverts or CV writing advice. We let the annotators carry out the filtering process of only choosing genuine CVs of software developers and programmers for annotation and reject but record any documents that did not fit this category. The annotators rejected 99 files as being either not CVs at all (49) or being out-of-domain CVs from other types of professionals (50). Therefore, just over 50% of the documents in the pool were used up during the annotation process.

3.2 Document Preparation

Before annotation, all candidate CVs were then automatically converted from Word DOC format to OpenOffice ODT as well as to Acrobat PDF format in a batch process using OpenOffice macros. The resulting contents.xml files for each ODT version of the documents contain the textual information of the original CVs. An XSLT stylesheet was used to simplify this format to a simpler in-house XML format, as the input into our pre-processing pipeline. We retained all formatting and style information in span elements for potential later use.

The pre-processing includes tokenization, sentence boundary detection, part-of-speech tagging, lemmatization, chunking, abbreviation detection and rule-based NER for person, location names and dates. This information extraction system is a modular pipeline built around the LT-XML2[4] and LT-TTT2[5] toolsets. The NER output is stored as standoff annotations in the XML. These pre-processed files were used as the basis for annotation.

3.3 Annotation Tool

For annotating the text of the CVs we chose MMAX2, the Java-based open source tool (Müller and Strube, 2006).[6] MMAX2 supports multiple levels of annotation by way of stand-off annotation. As a result MMAX2 creates one separate file for each level of annotation for each given base data file. Only the annotation level files get edited during the annotation phase. The base data files which contain the textual information of the documents do not change. In our project, we were interested in three levels of annotation, one for named entities (NEs), one for zones and one for relations between NEs. The MMAX2 GUI allows annotators to mark up nested structures as well as intra- and inter-sentential relations. Both of these functionalities were crucial to our annotation effort.

As the files used for annotation already contained some NEs which were recognized automatically using the rule-based NER system and stored in standoff XML, the conversion into and out of the MMAX2 format was relatively straightforward. For each file to be annotated, we created one base file containing the tokenized text and one entity file containing the rule-based NEs.[7]

3.4 Annotation Phases

We employed 3 annotators with various degrees of experience in annotation and computer science and therefore familiar with software engineering skills and terminology. The lead researcher of the project, the first author of this paper, managed the annotators and organized regular meetings with them.

We followed the agile corpus creation approach and carried out cycles of annotations, starting with a simple paper-based pilot annotation. This first annotation of 10 documents enabled us to get a first impression of the type of information contained in CVs of software engineers and programmers as well as the type of information we wanted to capture in the manual and automatic annotation. We drew up a first set of potential types of zones that occur within

[4] http://www.ltg.ed.ac.uk/software/ltxml2

[5] http://www.ltg.ed.ac.uk/software/lt-ttt2

[6] http://mmax2.sourceforge.net

[7] For more information on how this is done see Müller and Strube (2006).

CVs and the types of NEs that can be found within each zone (e.g. an EDUCATION zone containing NEs of type LOC, ORG and QUAL).

Using this set of potential markables, we decided on a subset of NEs and zones to be annotated in future rounds. Regarding the zones, we settled on annotating zone titles in a similar way as NEs. Our assumption was that recognizing the beginning of a zone can sufficiently identify zone boundaries. We did not include relations between NEs at this stages, as we wanted to get a clearer idea of the definitions of relevant NEs first before proceeding to relations.

We then carried out a second pilot annotation using 10 more CVs selected from the candidate pool. We used the revised annotation scheme and this time the annotation was done electronically using MMAX2. The annotators also had access to the PDF and DOC versions of each file in case crucial structural or formatting information was lost in the conversion. Files were annotated for NEs and zone titles. We also asked the annotators to answer the following questions:

- Does it make sense to annotate the proposed markables and what are the difficulties in doing so?
- Are there any interesting markables missing from the list?
- Are there are any issues with using the annotation tool?

Half way through the second pilot we scheduled a further meeting to discuss their answers, addressed any question, comments or issues with regard to the annotation and adjusted the annotation guidelines accordingly. At this point, as we felt that the definitions of NEs were sufficiently clear and added guidelines for annotating various types of binary relations between NEs, for example a LOC-ORG relation referring to a particular organization situated at a particular location, e.g. Google - Dublin. We list the final set of markables as defined at the end of the annotation process in Tables 2 and 3.

During the second half of the second pilot we asked the annotators to time their annotation and established that it can take between 30 minutes and 1.5 hours to annotate a CV. We then calculated pairwise IAA for two doubly annotated files which allowed us to get some evidence for which definition of NEs, zone titles and relations were still ambiguous or not actually relevant.

In parallel with both pilots, we also liaised closely with a local recruitment company to gain a first-hand understanding of what information recruiters are interested in when matching candidates to employments or employers. This consultation as well as the conclusions made after the second pilot led to further adaptions of the annotation scheme before the main annotation phase began.

Based on the feedback from the second pilot annotation, we also made some changes to the data conversion and the annotation tool setup to reduce the amount of work for annotators but without restricting the set of markables. In the case of some nested NEs, we propagated relations between embedded NEs that could be referred from the relations of the containing NEs. For example, two DATE entities nested within a DATERANGE entity, the latter of which the annotator related to an ORG entity, were related to the same ORG entity automatically. We also introduced a general GROUP entity which could be used by the annotators to mark up lists of NEs, for example, if they were all related to a different NE mention of type X. In that case, the annotators only had to mark up a relation between the GROUP and X. All implicit relations between the NEs nested in the GROUP and X were propagated during the conversion from the MMAX2 format back into the in-house XML format. This proved particularly useful for annotating relations between SKILL entities and other types of NEs.

Once those changes had been made, the main annotation phase began. Each in-domain CV that was loaded into the annotation tool already contained some NEs pre-annotated by the rule-based NER system (see Section 3.2). The annotators had to correct the annotations in case they were erroneous. Overall, the annotators reported this pre-annotation to be useful rather than hindering as they did not have to do too many corrections. At the end of each day, the annotators checked in their work into the project's subversion (SVN) repository. This provided us with additional control and backup in case we needed to go back to previous versions at later stages.

The annotation guidelines still evolved during the main annotation. Regular annotation meetings were

held in case the annotators had questions on the guidelines or if they wanted to discuss specific examples. If a change was made to the annotation guidelines, all annotators were informed and asked to update their annotations accordingly. Moreover, IAA was calculated regularly on sub-sections of the doubly annotated data. This provided more empirical evidence for the types of markables the annotators found difficult to mark up and where clarifications where necessary. The reasons for this were that their definitions were ambiguous or underspecified.

We deliberately kept the initial annotation scheme simple. The idea was for the annotators to shape the annotation scheme based on evidence in the actual data. We believe that this approach made the data set more useful for its final use to train and evaluate TM components. As a result of this agile annotation approach, we became aware of any issues very early on and were able to correct them accordingly.

4 Annotation Scheme

In this section, we provide a summary of the final annotation scheme as an overview of all the markables present in the annotated data set.

4.1 Named Entities

In general, we asked the annotators to mark up every mention of all NE types throughout the entire CV, even if they did not refer to the CV owner. With some exceptions (DATE in DATERANGE and LOC or ORG in ADDRESS), annotators were asked to avoid nested NEs and aim for a flat annotation. Discontinuous NEs in coordinated structures had to be marked as such, i.e. the NE should only contain strings that refer to it. Finally, abbreviations and their definitions had to be annotated as two separate NEs. The NE types in the final annotation guidelines are listed in Table 2. While carrying out the NE annotation, the annotators were also asked to set the NE attribute of type CANDIDATE (by default set to `true`) to `false` if a certain NE was not an attribute of the CV owner (e.g. the ADDRESS of a referee).

4.2 Zone Titles

Regarding the zone titles, we provided a list of synonyms for each type as context (see Table 2). The annotators were asked only to annotate main zone titles, ignoring sub-zones. They were also asked to mark up only the relevant sub-string of the text referring to the zone title and not the entire title if it contained irrelevant information.

Entity Type	**Description**
ADDRESS	Addresses with streets or postcodes.
DATE	Absolute (e.g. 10/04/2010), underspecified (e.g. April 2010) or relative dates (e.g. to date) including DATE entities within DATERANGE entities.
DATERANGE	Date ranges with a specific start and end date including ones with either point not explicitly stated (e.g. since 2008).
DOB	Dates of birth.
EMAIL	Email addresses.
JOB	Job titles and roles referring to the official name a post (e.g. software developer) but not a skill (e.g. software development).
LOC	Geo-political place names.
ORG	Names of companies, institutions and organizations.
PER	Person names excluding titles.
PHONE	Telephone and fax numbers.
POSTCODE	Post codes.
QUAL	Qualifications achieved or working towards.
SKILL	Skills and areas of expertise incl. hard skills (e.g. Java, C++, French) or general areas of expertise (e.g. software development) but not soft or interpersonal skills (e.g. networking, team work).
TIMESPAN	Durations of time (e.g. 7 years, 2 months, over 2 years).
URL	URLs
GROUP	Dummy NE to group several NEs for annotating multiple relations at once. The individual NEs contained within the group still have to be annotated.

Zone Title Type	**Synonyms**
EDUCATION	Education, Qualifications, Training, Certifications, Courses
SKILLS	Skills, Qualifications, Experience, Competencies
SUMMARY	Summary, Profile
PERSONAL	Personal Information, Personal Data
EMPLOYMENT	Employment, Employment History, Work History, Career, Career Record
REFERENCES	References, Referees
OTHER	Other zone titles not covered by this list, e.g. Publications, Patents, Grants, Associations, Interests, Additional.

Table 2: The types of NEs and zone titles annotated.

4.3 Relations

The binary relations that were annotated (see Table 3) always link two different types of NE mentions. Annotators were asked to mark up relations within the same zone but not across zones.

Relation Type	Description
TEMP-SKILL	A skill related to a temporal expression (e.g. Java - 7 years). TEMP includes any temporal NE types (DATE, DATERANGE and TIMESPAN).
TEMP-LOC	A location related to a temporal expression (e.g. Dublin - summer 2004).
TEMP-ORG	An organization related to a temporal expression (e.g. Google - 2001-2004).
TEMP-JOB	A job title related to a temporal expression (e.g. Software Engineer - Sep. 2001 to Jun. 2004).
TEMP-QUAL	A qualification related to a temporal expression (e.g. PhD - June 2004).
LOC-ORG	An organization related to a location (e.g. Google - Dublin).
LOC-JOB	A job title related to a location (e.g. Software Engineer - Dublin).
LOC-QUAL	A qualification related to a location (e.g. PhD - Dublin).
ORG-JOB	A job title related to an organization (e.g. Software Engineer - Google).
ORG-QUAL	A qualification related to an organization (e.g. PhD - University of Toronto).
GROUP-X	A relation that can be assigned in case a group of NEs all relate to another NE X. GROUP-X can be any of the relation pairs mentioned in this list.

Table 3: The types of relations annotated.

5 Inter-Annotator Agreement

We first calculated pairwise IAA for all markables at the end of the 2nd pilot and continued doing so throughout the main annotation phase. For each pair of annotations on the same document, IAA was calculated by scoring one annotator against another using precision (P), recall (R) and F_1.[8] An overall IAA was calculated by micro-averaging across all annotated document pairs.[9] We used F_1 rather than the Kappa score (Cohen, 1960) to measure IAA as the latter requires comparison with a random baseline, which does not make sense for tasks such as NER.

Table 4 compares the IAA figures we obtained for 2 doubly annotated documents during the 2nd pilot phase, i.e. the first time we measured IAA, to those we obtained on 9 different files once the main annotation was completed. For NEs and zone titles, IAA was calculated using P, R and F_1, defining two mentions as equal if they had the same left and right boundaries and the same type. Although this comparison is done over different sub-sets of the corpus, it is still possible to conclude that the NE IAA improved considerably over the course of the annotation process.

The IAA scores for the majority of NEs were increased considerably at the end, with the exception of SKILL for which the IAA ended up being slightly lower as well as DOB and PER of which there are not sufficient examples in either sets to obtain reliably results.[10] There are very large increases in IAA for JOB and ORG entities, as we discovered during the pilot annotation that the guidelines for those markables were not concrete enough regarding their boundaries and definitions. Their final IAA figures show that both of these types of NEs were still most difficult to annotate at the end. However, a final total IAA of 84.8 F_1 for all NEs is a relatively high score. In comparison, the final IAA score of 97.1 F_1 for the zone titles shows that recognizing zone titles is an even easier task for humans to perform compared to recognizing NEs.

When calculating IAA for relations, only those relations for which both annotators agreed on the NEs were included. This is done to get an idea of the difficulty of the RE task independently of NER. Relation IAA was also measured using F_1, where relations are counted as equal if they connect exactly the same NE pair. The IAA for relations between NEs within CVs is relatively high both during the pilot annotation and at the end of the main annotation and only increased slightly over this time. These figures show that this task is much easier than annotating relations in other domains, e.g. in biomedical research papers (Alex et al., 2008a).

The IAA figures show that even with cyclic annotation, evolving guidelines and continuous updating, human annotators can find it challenging to annotate some markables consistently. This has an effect on the results of the automatic annotation where the annotated data is used to train ML-based models and to evaluate their performance.

[8] P, R and F_1 are calculated in standard fashion from the number of true positives, false positives and false negatives.

[9] Micro-averaging was chosen over macro-averaging, since we felt that the latter would give undue weight to documents with fewer markables.

[10] The reason why there are no figures for POSTCODE and TIMESPAN entities for the pilot annotation is that none appeared in those documents.

	(1) 2nd Pilot Annotation				(2) End of Main Annotation				(3) Automatic Annotation			
Type	P	R	F_1	TPs	P	R	F_1	TPs	P	R	F_1	TPs
Named Entities												
ADDRESS	100.0	100.0	100.0	1	100.0	100.0	100.0	10	13.8	16.0	14.8	8
DATE	62.5	92.6	74.6	25	98.5	98.5	98.5	191	94.1	95.7	94.9	1,850
DATERANGE	91.3	95.5	93.3	21	98.6	97.3	97.9	71	91.4	87.0	89.2	637
DOB	100.0	100.0	100.0	1	75.0	100.0	85.7	3	70.8	70.8	70.8	17
EMAIL	100.0	100.0	100.0	2	100.0	100.0	100.0	8	95.9	100.0	97.9	93
JOB	39.1	52.9	45.0	9	72.5	69.9	71.2	95	70.5	61.4	65.6	742
LOC	88.9	100.0	94.1	16	100.0	95.8	97.9	137	83.2	87.3	85.2	1,259
ORG	68.0	81.0	73.9	17	93.4	86.4	89.8	171	57.1	44.7	50.2	749
PER	100.0	100.0	100.0	2	100.0	95.0	97.4	19	69.8	40.5	51.2	196
PHONE	100.0	100.0	100.0	4	100.0	100.0	100.0	16	90.9	85.7	88.2	90
POSTCODE	-	-	-	-	90.9	90.9	90.9	10	98.3	71.3	82.6	57
QUAL	9.1	7.7	8.3	1	68.4	81.3	74.3	13	53.9	27.2	36.1	56
SKILL	76.6	86.8	81.4	210	79.3	79.0	79.2	863	67.9	66.5	67.2	5,645
TIMESPAN	-	-	-	-	91.7	91.7	91.7	33	74.0	76.8	75.4	179
URL	100.0	100.0	100.0	2	100.0	100.0	100.0	43	97.2	90.5	93.7	209
All	73.0	84.1	78.1	311	85.4	84.2	84.8	1,683	73.5	69.4	71.4	11,787
Zone Titles												
EDUCATION	100.0	100.0	100.0	3	100.0	100.0	100.0	9	86.3	75.0	80.3	63
EMPLOYMENT	100.0	100.0	100.0	1	100.0	88.9	94.1	8	83.1	69.7	75.8	69
OTHER	100.0	100.0	100.0	1	-	-	-	-	39.3	28.2	32.8	22
PERSONAL	25.0	25.0	25.0	1	100.0	100.0	100.0	4	65.4	53.1	58.6	17
REFERENCES	100.0	100.0	100.0	1	100.0	100.0	100.0	3	94.4	89.5	91.9	17
SKILLS	33.3	40.0	36.4	2	100.0	100.0	100.0	7	63.8	38.9	48.4	44
SUMMARY	-	-	-	-	75.0	100.0	85.7	3	82.2	64.9	72.6	37
All	56.3	60.0	58.1	9	97.1	97.1	97.1	34	72.7	55.8	63.2	269
Relations												
DATE-JOB	-	-	-	-	100.0	83.3	90.9	10	28.1	44.7	34.5	110
DATE-LOC	-	-	-	-	88.9	72.7	80.0	8	71.3	52.7	60.6	223
DATE-ORG	-	-	-	-	100.0	88.2	93.8	15	53.0	51.5	52.3	218
DATE-QUAL	-	-	-	-	100.0	100.0	100.0	6	60.6	73.1	66.3	57
DATERANGE-JOB	77.8	100.0	87.5	7	91.7	100.0	95.7	66	80.4	72.5	76.2	663
DATERANGE-LOC	91.7	100.0	95.7	11	85.4	79.6	82.4	70	82.0	82.7	82.4	735
DATERANGE-ORG	93.8	100.0	96.8	15	80.2	76.2	78.2	77	72.2	76.4	74.2	644
DATERANGE-QUAL	100.0	100.0	100.0	1	100.0	100.0	100.0	21	71.1	62.1	66.3	59
DATERANGE-SKILL	89.0	98.1	93.3	105	82.2	100.0	90.5	352	61.1	33.7	43.4	1,574
DATE-SKILL	100.0	9.1	16.7	1	95.0	67.1	78.6	57	23.6	54.5	33.0	368
JOB-LOC	NaN	0.0	NaN	0	91.8	65.6	76.5	78	77.0	69.1	72.8	932
JOB-ORG	87.5	100.0	93.3	7	86.8	73.3	79.5	99	64.6	50.7	56.8	758
JOB-TIMESPAN	-	-	-	-	85.7	54.6	66.7	6	56.0	61.8	58.8	47
LOC-ORG	NaN	0.0	NaN	0	89.6	71.4	79.5	120	79.7	78.9	79.3	1,044
LOC-QUAL	NaN	0.0	NaN	0	100.0	100.0	100.0	19	75.6	78.7	77.1	133
LOC-TIMESPAN	-	-	-	-	100.0	75.0	85.7	3	48.2	36.1	41.3	13
ORG-QUAL	NaN	0.0	NaN	0	95.2	95.2	95.2	20	77.8	71.4	74.5	140
ORG-TIMESPAN	-	-	-	-	83.3	55.6	66.7	5	55.9	33.3	41.8	19
SKILL-TIMESPAN	-	-	-	-	86.1	74.0	79.6	37	59.5	52.6	55.8	280
All	85.5	83.1	84.2	147	86.8	82.6	84.6	1,069	63.1	55.3	59.0	8,017

Table 4: IAA for NEs, zone titles and relations in precision (P), recall (R) and F_1 at two stages in the annotation process: (1) at the end of the second pilot annotation and (2) at the end of the main annotation phase; as well as automatic annotation scores (3) on the blind TEST set. The total number of true positives (TPs) is shown to provide an idea of the quantities of markables in each set.

6 Automatic Annotation

Table 4 also lists the final scores of the automatic ML-based NER and RE components (Alex et al., 2008b) which were adapted to the recruitment domain during the TXV project. Following agile methods, we trained and evaluated models very early into the annotation process. During the system optimization, learning curves helped to investigate for which markables having more training data available would improve performance.

The NER component recognizes NEs and zone titles simultaneously with an overall F_1 of 71.4 (84.2% of IAA) and 63.2 (65.0% of IAA), respectively. Extremely high or higher than average scores were obtained for DATE, DATERANGE, EMAIL, LOC, PHONE, POSTCODE, TIMESPAN and URL entities. Mid-range to lower scores were obtained for ADDRESS, DOB, JOB, ORG, PER, QUAL and SKILL entities. One reason is the similarity between NE types, e.g. DOB is difficult to differentiate from DATE. The layout of CVs and the lack of full sentences also pose a challenge as the NER component is trained using contextual features surrounding NEs that are often not present in CV data. Finally, the strict evaluation counts numerous boundary errors for NEs which can be considered correct, e.g. the system often recognizes organization names like "Sun Microsystems, Inc" whereas the annotator included the full stop at the end ("Sun Microsystems, Inc.").

The RE component (Haddow, 2008) performs with an overall F_1 of 59.0 on the CV TEST set (69.7% of IAA). It yields high or above average scores for 10 relation types (DATE-LOC, DATE-QUAL, DATERANGE-JOB, DATERANGE-LOC, DATERANGE-ORG, DATERANGE-QUAL, JOB-LOC, LOC-ORG, LOC-QUAL, ORG-QUAL). It yields mid-range to low scores for the other relation types (DATE-JOB, DATE-ORG, DATERANGE-SKILL, DATE-SKILL, JOB-ORG, JOB-TIMESPAN, LOC-TIMESPAN, ORG-TIMESPAN, SKILL-TIMESPAN). The most frequent type is DATERANGE-SKILL, a skill obtained during a particular time period. Its entities tend to be found in the same zone but not always in immediate context. Such relations are inter-sentential, i.e. their entities are in different sentences or what is perceived as sentences by the system. Due to nature of the data, there are few intra-sentential relations, relations between NEs in the same sentence. The further apart two related NEs are, the more difficult it is to recognize them. Similarly to NER, one challenge for RE from CVs is their diverse structure and formatting.

7 Discussion and Conclusion

The increase in the IAA figures for the markables over time show that agile corpus annotation resulted in more qualitative annotations. It is difficult to prove that the final annotation quality is higher than it would have been had we followed the traditional way of annotation. Comparing two such methods in parallel is very difficult to achieve as the main aim of annotation is usually to create a corpus and not to investigate the best and most efficient method.

However, using the agile approach we identified problems early on and made improvements to the annotation scheme and the setup during the process rather than at the end. Given a fixed annotation time frame and the proportion of time we spent on correcting errors throughout the annotation process, one might conclude that we annotated less data than we may have done, had we not followed the agile approach. However, Voormann and Hut (2008) argue that agile annotation actually results in more useable data at the end and in less data being thrown away.

Had we followed the traditional approach, we would unlikely have planned a correction phase at the end. The two main reason for that are cost and the general belief that the more annotated data the better. A final major correction phase is usually viewed as too expensive during an annotation project. In order to avoid this cost, the traditional approach taken tends to be to create a set of annotation guidelines when starting out and hold off the main annotation until the guidelines are finalized and considered sufficiently defined. This approach does not lend itself well to changes and adjustments later on which are inevitable when dealing with natural language. As a result the final less accurate annotated corpus tends to be accepted as the ground truth or gold standard and may not be as suitable and useful for a given purpose as it could have been following the agile annotation approach. Besides changing the way in which annotators work, we recognize the need for more flexible annotation tools that allow annotators to implement changes more rapidly.

References

Beatrice Alex, Malvina Nissim, and Claire Grover. 2006. The impact of annotation on the performance of protein tagging in biomedical text. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Bea Alex, Claire Grover, Barry Haddow, Mijail Kabadjov, Ewan Klein, Michael Matthews, Richard Tobin, and Xinglong Wang. 2008a. The ITI TXM corpora: Tissue expressions and protein-protein interactions. In *Proceedings of the Workshop on Building and Evaluating Resources for Biomedical Text Mining at LREC 2008*, Marrakech, Morocco.

Beatrice Alex, Claire Grover, Barry Haddow, Mijail Kabadjov, Ewan Klein, Michael Matthews, Richard Tobin, and Xinglong Wang. 2008b. Automating curation using a natural language processing pipeline. *Genome Biology*, 9(Suppl 2):S10.

Sue Atkins, Jeremy Clear, and Nicholas Ostler. 1992. Corpus design criteria. *Literary and Linguistic Computing*, 7(1):1–16.

Douglas Biber. 1993. Representativeness in corpus design. *Literary and Linguistic Computing*, 8(4):243–257.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Barry Haddow. 2008. Using automated feature optimisation to create an adaptable relation extraction system. In *Proceedings of BioNLP 2008*, Columbus, Ohio.

Christoph Müller and Michael Strube. 2006. Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy. New Resources, New Tools, New Methods.*, pages 197–214. Peter Lang, Frankfurt. (English Corpus Linguistics, Vol.3).

Václav Novák and Magda Razímová. 2009. Unsupervised detection of annotation inconsistencies using apriori algorithm. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 138–141, Suntec, Singapore.

Winston Royce. 1970. Managing the development of large software systems. In *Proceedings of IEEE WESCON*, pages 1–9.

Holger Voormann and Ulrike Gut. 2008. Agile corpus creation. *Corpus Linguistics and Linguistic Theory*, 4(2):235–251.