Information Extraction via Double Classification

An De Sitter
Dept. of Mathematics and Computer Science
University of Antwerp
Middelheimlaan 1
2020 Antwerpen, Belgium
anneleen.desitter@ua.ac.be

Walter Daelemans
CNTS
University of Antwerp
Universiteitsplein 1
2610 Antwerpen, Belgium
daelem@uia.ua.ac.be

Abstract

Information Extraction is concerned with extracting relevant information from a (collection of) documents. We propose an approach consisting of two classification-based machine learning loops. In a first loop we look for the relevant sentences in a document. In the second loop, we perform a word-level classification. We test the system on the Software Jobs corpus and we do an extensive evaluation in which we discuss the influence of the different parameters. Furthermore we show that the type of evaluation method has an important influence on the results.

1 Introduction

An Information Extraction (IE) system has as goal to extract relevant information from a (collection of) document(s). What kind of information is relevant is defined by a template.

In this paper we propose an approach consisting of two classification-based machine learning loops. In the first loop we select the sentences in the document that might contain relevant information. In a second loop we perform a deeper analysis of those relevant sentences by performing word-level classification. By using a rule-based classifier (namely Ripper [3]) for the second loop (unlike a Naive Bayes classifier for the first loop), we obtain human-readable rules.

We did an extensive evaluation of this Double Classification approach. To achieve this, we adapted the evaluation described by Freitag [4]. We test our method on the Software Jobs corpus [1] and study the influence of the parameter settings on the one hand, and the evaluation methodology on the other hand. We obtain good results on most of the template slots, although for some slots (`language` and `area`) the word-classification approach is not yet capable of finding good rules. Excluding those two slots, we obtain a recall of 77% and a precision of up to 59% if we require that *all occurrences* of an item are extracted. If we use the *one best per document* approach, we obtain recall up to 77% and precision up to 65%. Other IE systems that are evaluated on this corpus (RAPIER [2] and WHISK [12]) obtain a better precision but a worse recall. We also show that the double classification approach indeed improves upon a single word-based classification approach.

The main contributions of this paper are the introduction of the double classification method on the one hand, and on the other hand, an extensive evaluation in which we show that the method of evaluation has an important influence on the results, and therefore on any comparison between systems and approaches. The organization of the paper is as follows: in section 2 we introduce the double classification approach. Section 3 discusses the need for a clear evaluation methodology, section 4 shows the experimental results. Finally, section 5 gives conclusions, related research and further work.

2 Double Classification

The intuition for our approach comes from the observation that humans, when looking for a specific piece of information in a document, don't read the whole document in detail at once. One starts reading globally and based on a quick and superficial analysis then starts reading parts of the text in more depth. It is only in the second phase that the information to be extracted is identified.

In practice, we divide each document into sen-

[1] Downloaded from ftp:/ftp.cs.utexas.edu/pub/mooney/job-data/job600.tar.gz

tences. For each sentence we use a Naive Bayes classifier working on a bag-of-words representation of the sentence to decide wether it is relevant or not. By doing this, we get a set of relevant sentences $s1$, $s2$, ...per document ordered by a measure of certainty. In a second step, we use a rule-based classifier that decides for each word in the sentence whether it belongs to the entity to be extracted. We can do this for the best, the two best,... or all relevant sentences.

Example 1 *Figure 1 shows an example of a job advertisement from the Software Jobs corpus. The item to be extracted is e.g. the job title. Suppose that a Naive Bayes classifier considers three sentences as relevant, thus having a confidence higher than 0.5. In the second step we may use the best relevant sentence, the two best or all the relevant sentences to do the word-level classification. In section 4 we discuss the consequences of each choice.*

The main motivation for this approach is that the first loop helps to solve the unbalanced training data problem. Suppose we have 100 documents of 40 sentences each. A sentence consists of 10 words. Every document has exactly 1 sentence containing a three-word slot filling. Without the selection of sentences in the first loop, only 0.8% of the words belong to the positive class. Because of this skewed class distribution, a word-based classification approach is infeasible. In our approach however, with the selection of resp. 1 or 3 sentences in each loop, the sizes of the positive classes become respectively 30% and 10%, hence enabling a word-based classification.

We now describe the two classification loops in more detail.

2.1 Sentence classification

In the first loop we want to decide at sentence-level whether a sentence contains relevant information or not. Because we are only doing a deeper analysis of those sentences that this step selects as relevant ones, we need a high recall. In addition, we want the precision to be as high as possible, but this is less crucial than the high recall. For the sentence classification loop, we use a Naive Bayes classifier. We used `Bow` [9], a library of C-code designed and written by Andrew McCallum, and in particular we used `Rainbow`, a front-end that does document classification. `Rainbow` first reads all training data and archives a model containing their statistics. Using this model, `Rainbow` performs classification for each test document.

In this case, each sentence is seen as a separate document. We use the sentences as bags of words without adding any further linguistic information such as POS tags.

The greatest problem we encountered was the large difference between the number of relevant and non-relevant sentences. In our experiments (see section 4) we typically have 1 relevant sentence for each 40 non-relevant ones. This leads to highly unbalanced classes with which standard Naive Bayes has difficulties. As an initial solution, we sample from the set of non-relevant sentences about the same amount of data as we have relevant sentences. For testing purposes we use the data as is (with unbalanced classes). By doing this, we achieve our goal of high recall (we obtain 95-100% recall) and precision of 80-90% at sentence level. So, almost all relevant sentences are recognized, and not too many mistakes are made on the non-relevant sentences.

2.2 Word Classification

The word-level classification is developed in analogy with (NP) chunking. Chunking groups words in sentences into coherent groups such as Nominal Phrases (NPs, e.g. "the old man) and Adverbial Phrases (AdvP, e.g. "not very well). [11] described chunking as a tagging task to be solved with transformation-based error-driven learning. Therefore each word is classified as *in*, *out*, or *at the border* of a particular chunk. In analogy to this, we classify each word of the sentence as being *in* or *out* of the entity to be extracted.

Example 2 *If the entity to be extracted is the* job title*, sentence 13 from the advertisement in Figure 1 should be annotated as follows (* $_I$ *stands for* in*,* $_O$ *for* out *the entity):* $Major_O$ $corporations_O$ $have_O$ $immediate_O$ $openings_O$ for_O $junior_O$ $database_I$ $administrators_I$ $._O$ *In this representation 'database administrators' is unambiguously annotated as the job title to be extracted.*

For the word-level classification, we prefer having human-readable rules, such that we have some insight into how the task has been solved. We used RIPPER [3], a rule-based classification algorithm.

To keep the system easily adaptable to other tasks, we tried to use as little information as possible to solve the task. The default feature vector (per word) consists of the word to be classified, its POS tag, a set of the three words before, a set of the three POS tags before, a set of the three words after, and a set of the three POS tags after the word. We made following variations on this default:

class-before We added the class of the former word. For training we used the actual class of the former word, for testing the predicted class. By doing this, we simulate a simple HMM-approach without losing the advantage of having readable rules.

context Instead of using only the immediate context of a word, we use the whole sentence (words and POS tags), still divided in a set of words (tags) before and a set of words (tags) after the actual word.

attributes We tried adding a few different extra attributes: place in the sentence, and boolean values which indicate if the word starts with a capital, consists of all capitals or contains a digit.

Example 3 *For sentence 13 from the job advertisement shown in Figure 1 the* class-before *feature vector for the word 'developers' is :*
WORDS-BEFORE: for Visual Basic
TAGS-BEFORE: IN JJ NNP
WORDS-AFTER: .
TAGS-AFTER: Punc
WORD: developers
TAG: NNS
CLASS-BEFORE: the prediction on the former word.

The word-level classifier was trained only on the relevant sentences.

Figure 2 shows an example of a hypothesis built by RIPPER for extracting `salary` from the Software Jobs corpus. The first rule, for example, can be read as follows: if the class of the word before was 'I', the word itself is '$' and the tag after the word is 'CD', the current word is tagged as 'I'.

3 Evaluation

For the evaluation of IE systems, typically standard metrics from Information Retrieval are used, namely precision and recall. Precision is the number of correctly predicted entities divided by all entities predicted. Recall is the number of correctly predicted entities divided by the number of entities that should have been found. To compute those and other metrics, we construct a confusion matrix.

In Information Retrieval or in standard classification applications, a confusion matrix is straightforward to build: a document is relevant or not, a mushroom is poisoned or not,... However, in IE the situation is not always as straightforward. Even with manually annotating a text, different annotators will often disagree.

Example 4 *Consider the following sentence:*
`Our company is looking for a senior database administrator (male/ female) to lead the team.`
there are different possibilities for `job title`*:* `database administrator`*,* `senior database administrator`*,* `database administrator (male/female)`*, and* `senior database administrator (male/female)` *can all be seen as valid answers by different annotators.*

As a result of this, researchers have used various criteria for counting an extracted item as being correct. Some require the exact same boundaries as were manually annotated in the training set, others consider "almost the same" boundaries sufficient.

What criterion has to be used, may depend on the situation as well. If the goal of the IE system is to fill a database on which queries can be asked, one wants to be sure that whatever gets into the database is exact. On the other hand, if someone has as task to point out the job title in each document in a set of job advertisements, he will be helped more by a system that gives a solution overlapping with the correct answer for each document, than by one that returns the exact answer for only a small portion of the documents.

However, to be able to interpret results and to compare different IE systems, it is important to know *how* one has performed the evaluation. To define an evaluation, the first choice that has to be made, is whether we want to find *all occurrences* (AO) of an entity (e.g. every mention of the job title in the advertisements should be found), or whether it suffices to find one occurrence for each template slot. The latter approach is called *one best per document* (OBD).

On the other hand, we have to decide when an extracted entity is counted as a *true positive.* In order to implement the ideas mentioned before, Freitag proposed three basic criteria [4]:

exact The predicted instance matches exactly an actual instance.

contain The predicted instance strictly contains an actual instance, and at most k neighboring tokens.

overlap The predicted instance overlaps an actual instance, there are at most k neighboring tokens, and maximum l missing tokens.

We implemented those criteria as three operators, the latter two having 1 (k) and 2 (k,l) parameters.

Example 5 *Suppose the sentence in Example 4 is annotated as follows:*
`Our company is looking for a <title> senior database administrator </title> (male/female) to lead the team.`

- `senior database administrator` *is correct for the three operators;*
- `senior database administrator (male/female)` *is wrong when using the* exact *operator, is correct using the* contain *and* overlap *operators admitting 1 neighboring token.*
- `database administrator (male/female)` *is wrong when using the* exact *or* contain *operator, and is correct when using the* overlap *operator and admitting 1 neighboring token and 1 missing token.*

In our experiments, we show that the choice between AO and OBD on the one hand, and the type of operator on the other hand, often has a large influence on the results obtained. Thus, it is of utmost importance that the evaluation method is clearly indicated, to be able to interpret the results correctly.

4 Experiments

4.1 Setup

For our experiments we used the Software Jobs corpus: a set of 600 computer-related job postings. Figure 1 shows an example of a job advertisement from the corpus. The templates for this corpus consist of 17 slots: `id`, `country`, `state`, `city`, `company`, `title`, `salary`, `recruiter`, `post-date`, `desired years of experience`, `required years of experience`, `desired degree`, `required degree`, `platform` (e.g. Windows NT), `application` (e.g. SQL server), `area` and `language` (e.g. Java)). Several of those slots take multiple fillers (e.g. language, area, ...). Not all entities appear in each document.

For our experiments we used 10-fold cross-validation[2]. We report recall and precision, averaged over those cross-validations. For some overall results, we report F_1-measure as well. We evaluate the AO as well as the OBD setting. We evaluate each slot separately and report average recall and precision (and F_1-measure) over all slots as overall scores.

[2]Except for some rare slots as e.g. desired years of experience. In this case we used 5-fold cross-validation to omit large fluctuations in the scores.

feature vector	AO		OBD	
	recall	precision	recall	precision
class-before	77%	43%	74%	26%
attributes	78%	44%	75%	38%
small context	49%	46%	58%	40%

Table 1: Influence of the feature vector used to extract the "company". All relevant sentences were used, we report the scores obtained on finding the exact boundaries.

nr sentences	AO		OBD	
	recall	precision	recall	precision
1	18%	33%	33%	37%
2	34%	35%	34%	37%
3	40%	34%	35%	37%
all	43%	31%	35%	35%

Table 2: Influence of the number of relevant sentences used in the word-level classifier to extract the "title". The feature vector containing the class-before was used, we report the scores obtained on finding the exact boundaries.

4.2 Results

We report results on two fields: first we evaluate the influence of different parameter settings of the double classification method on recall. Secondly, we report on how the different evaluation criteria influence the results on single slots and on the overall result. Finally, we did some testing with using only the word-level classifier, and show that performance suffers a lot from omitting the sentence-level classifier.

Influence of the parameter settings

The double classification method has two important parameters: the feature vector used for the word-level classification and the number of sentences indicated as relevant by the sentence-level classifier that are used for the word-classifier.

Table 1 shows the influence of the feature vector used to extract `company` from the job advertisement. Early tests(not included in the table) showed that adding the (predicted) class of the former word

	AO		OBD	
	recall	precision	recall	precision
exact	43%	31%	35%	35%
contains(2)	54%	39%	44%	45%
overlaps(2,1)	84%	61%	70%	71%

Table 6: Influence of the operator used to evaluate the extraction of the "title". All relevant sentences were used, feature vector is default with class-before.

adds important information. The first row gives the scores when using the class-before feature vector. In the second row, we add extra attributes as described before. In the third row, more context is added to the class-before feature vector. We see that adding extra attributes does improve the precision somewhat in the OBD-case, but doesn't help otherwise. In this case of extracting 'company', adding extra context deteriorates the results as far as recall is concerned, while precision is improving the result somewhat. We tested this on other entities as well and noticed similar results.

Table 2 shows the influence of the number of relevant sentences that are passed from the sentence-level classifier to the word-level classifier, tested in particular for the 'title' slot. As might be expected, this has no real influence on recall or precision in the OBD setting, but it has a rather large impact on the recall in the AO setting. By using more sentences, we obtain a higher recall, but when using all relevant sentences, the precision decreases, although not too much.

Table 3 gives an overview of the scores of each template slot, requiring exact boundaries, using the class-before feature vector and all relevant sentences. RIPPER failed to find good rules for the slots `language` and `area` using the current feature vectors, and obtained no better results than the default error (on word-basis). One possible explanation is that those fields mostly take multiple slot fillers which typically are mentioned in the document as an enumeration of some sort. Regular expressions probably would do better for those entities. Further important remarks are:

1. Some entities, e.g. 'company' and 'title' obtain much better results using more lenient evaluation methods, as is discussed in the next paragraph.

2. 'state' as well as 'city' obtain a better precision but worse recall by using only the best relevant sentence for the word-classification instead of all relevant sentences.

Influence of evaluation methodology

In section 3 we discussed that a more lenient evaluation than requiring exact boundaries can be useful. We evaluated all entities with three operators: requiring exact boundaries (*exact*), admitting two neighboring tokens (*contain(2)*), and admitting two neighboring tokens plus missing at most one token (*overlap(2,1)*). Table 6 shows the results of those evaluations for the 'title' slot. Recall as well as precision increase a lot by using a more lenient operator. This gives us an important insight in the errors made by the IE system as well: the large gap between *exact* and *overlap*-scores show that the method is capable of finding "almost correct" solutions.

Although not all entities are influenced equally by this, we see important differences in the overall scores. Table 5 gives recall and precision for the overall score. The overall score has been calculated by taking the average over all slots, except for `language` and `area`, because for those results we did not obtain better results than the default error. Two other systems that have been trained on (subsets of) the same data set, RAPIER [2] and WHISK [12], both obtain recall of about 55% and precision of about 85%. They used different methods to count performance, so comparison is difficult, but it still seems clear that although we obtain a significant lower precision, our recall is much higher. A more extensive comparison using exactly the same evaluation method and data should be done to allow more reliable comparisons.

Single level classification

In Table 4 we compare our double classification approach with a baseline system which uses only the word-level classifier. For this baseline system we trained Ripper on approximately 250 (randomly selected) job advertisements, using all (positive as well as negative) sentences. We tested on approximately 60 (randomly selected) job advertisements. Because of memory requirements, it was not possible to use all available data. We did use the class-before setting. We tested the slots `country`, `state`, `salary`, `company` and `title`. The scores for `country` are comparable to the scores obtained by the double classification, `salary` gives worse, but still acceptable results, but the other fields give much worse results than obtained by the double classification approach. For `state` we see that the single classifier is not able to distinguish between the states that should be extracted and the other mentions of states. A deeper look at the results for `company` and `title` taught us that although the single classifier is able to find the 'almost' place of the entities (`title` obtained 35% precision, 37% recall for *all occurrences*, 30% precision, 27% recall for *one best per document* using the overlap(2,1) measure), it is not capable of finding the correct fillers.

5 Conclusions, related research and further work

We proposed a double classification loop approach to alleviate the problem of unbalanced training data sets in a classification-based learning approach to IE. An important advantage of this method is that by using a rule-based classifier for the second loop, we obtain human-readable extraction rules. We used no semantic information and only POS tags as syntactic information in order to keep the system as portable as possible. Furthermore we did an extensive evaluation in which we discussed the effect of different parameters of the system and showed that it is of utmost importance to be clear about the evaluation methodology used to be able to interpret the results correctly.

Although a hmm approach[1, 5, 6, 10] to IE also uses basically a classification-based approach with a state for words that are part of the filler of a slot to be extracted and a state for the 'empty tag', they are limited in the amount of lexical context they can take into account, and in the amount and types of information they can take into account without running into severe sparse data problems. In [13] it is shown that word-classification based learning methods can outperform hmms.

We believe the double classification approach is new in machine learning-based IE. The first, sentence-level, classification loop is similar to machine learning approaches to document summarization, such as [8], in which sentences are classified as relevant or not for appearing in the summary. A two-stage approach in the context of hybrid statistical and knowledge-based IE can be found in [7].

Future research concerns the improvement of the *double classification* approach by extracting different template slots at the same time. Furthermore, we are planning to test this IE system on other domains and compare it to alternative methods. We are also working on a formalization of the evaluation methodology for IE systems.

References

[1] D. Bikel et al. Nymble: a high-performance learning name-finder. In *Proc. ANLP*, 194–201, 1997.

[2] M. E. Califf and R. J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. In *http://citeseer.nj.nec.com/califf02bottomup.html.*

[3] W. Cohen. Fast effective rule induction. In *Proc. ICML*, 1995.

[4] D. Freitag. Machine learning for information extraction in informal domains. In *Phd thesis, Carnegie Mellon University, Pittsburgh PA.*, 1998.

[5] D. Freitag and A. McCallum. Information extraction using hmms and shrinkage. In *Proc. AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[6] D. Freitag and A. McCallum. Information extractaction with hmm structures learned by stochastic optimization. In *Proc. of AAAI*, 2000.

[7] P. Jacobs, G. Krupka, and L. Rau. Lexico-semantic pattern matching as a companion to parsing in text understanding. In *Fourth DARPA Speech and Natural Language Workshop*, 337–342, 1991.

[8] J. Kupiec, J. O. Pedersen, and F. Chen. A trainable document summarizer. In *ACM SIGIR*, 68–73, 1995.

[9] A. McCallum. Bow: a toolkit for statistical language modeling, text retrieval, classification and clustering. In *http://www.cs.cmu.edu/∼mccallum/bow*, 1996.

[10] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proc. ICML*, 2000.

[11] L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In *Proc. Workshop on Very Large Corpora, ACL*, 82–94, 1995.

[12] S. Soderland. Learning information extraction rules for semi-structured and free text. In *Machine Learning 34*, 233–272, 1999.

[13] J. Zavrel and W. Daelemans. Feature-rich memory-based classification for shallow nlp and information extraction. In *Text Mining. Theoretical aspects and applications. Springer LCNS series.*, 2003.

```
1.    From : spectrum@onramp.net
2.    Newsgroups : austin.jobs
3.    Subject : <country> US </country> - <state> TX </state> - <city> Austin
      </city>
      - <language> VISUAL BASIC </language> <title> Developers </title>
      <salary> 50Kto 70K </salary> Date : Sat , <post-date> 23 Aug 97
      </post-date> 09 : 52 : 21
4.    Organization : OnRamp Technologies , Inc. ; ISP
5.    Lines : 65
6.    Message - ID : < <id> NEWTNews.872347949.11738.consultsws - n </id> >
      NNTP - Posting - Host : ppp10 - 28.dllstx.onramp.net
      ...        ...        ...
10.   <country> US </country> - <state> TX </state> - <city> Austin </city>
      - junior <title> database administrators </title>
      <salary> 50Kto 70K </salary>
11.   POSTING I.D .
12.   D05
13.   Major corporations have immediate openings for junior
      <title> database administrators </title> .
14.   <req-years-experience> 2 </req-years-experience>
      - <desired-years-experience> 5 </desired-years-experience> years
      experience ; <application> Oracle </application> or <application> SQL
      Server </application> helpful .
15.   <platform> Windows 95 </platform> and <platform> Windows NT
      </platform> programming a plus .
16.   Please contact Bill Owens at ( 972 ) 484 - 9330 ; FAX ( 972 ) 243 - 0120 at
      <recruiter> Resource Spectrum </recruiter> .
      ...        ...        ...
26.   <recruiter> Resource Spectrum </recruiter>
27.   5050 Quorum Dr. , Ste 700
28.   Dallas , Texas 75240
      ...        ...        ...
```

Figure 1: (Part of) an annotated example of a job advertisement.

```
i :- CLASS_BEFORE=i, TAGS_AFTER   CD, WORD   '$'.
i :- CLASS_BEFORE=i, TAG   CD.
i :- CLASS_BEFORE=i, TAGS_AFTER   CD, WORDS_BEFORE   ':'.
i :- WORD   '$'.
i :- CLASS_BEFORE=i, TAGS_AFTER   CD, WORDS_AFTER   '000'.
i :- CLASS_BEFORE=i, TAG   NN, WORD   '-'.
i :- TAG   TO.
i :- CLASS_BEFORE=i, WORDS_AFTER   ':', WORDS_BEFORE   '$'.
i :- CLASS_BEFORE=i, WORD   '-'.
i :- CLASS_BEFORE=i, WORD   hr.
i :- CLASS_BEFORE=i, WORDS_BEFORE   '-', WORDS_AFTER   ';'.
i :- CLASS_BEFORE=i, WORD   '55K'.
i :- CLASS_BEFORE=i, TAG   NN, TAGS_AFTER   IN.
i :- WORDS_BEFORE   Salary, WORDS_BEFORE   ':', WORDS_AFTER   '-'.
i :- WORDS_BEFORE   in, WORDS_AFTER   '.'.
i :- CLASS_BEFORE=i, WORDS_AFTER   and.
i :- WORDS_AFTER   '70K', TAG   CD.
default o.
```

Figure 2: Hypothesis built by RIPPER for the 'salary' entry.

	AO			OBD		
	recall	precision	F_1	recall	precision	F_1
ID	98%	97%	97%	96%	99%	97%
country	98%	92%	95%	94%	91%	92%
state	97%	77%	86%	95%	93%	94%
city	95%	84%	89%	92%	90%	91%
company	78%	45%	57%	74%	26%	38%
title	43%	31%	36%	35%	35%	35%
salary	70%	56%	62%	72%	62%	67%
recruiter	79%	40%	53%	74%	44%	55%
post-date	99%	84%	91%	97%	99%	98%
desired degree	45%	28%	35%	37%	29%	33%
required degree	43%	29%	35%	51%	41%	45%
desired years experience	55%	33%	41%	66%	36%	47%
required years experience	80%	50%	62%	81%	72%	76%
platform	34%	31%	32%	38%	35%	36%
application	29%	32%	30%	30%	31%	30%
area	17%	16%	16%	18%	16%	17%
language	27%	25%	26%	34%	33%	33%
overall	63.9%	49.9%	55.5%	63.7%	54.8%	58.0%

Table 3: Individual results for each entity. All relevant sentences were used, feature vector is default with class-before, exact boundaries were required.

	AO			OBD		
	single	double best relevant	double all relevant	single	double best positive	double all positive
company	6%/4%	41%/22%	49%/77%	0%/0%	40%/80%	26%/74%
country	87%/92%	77%/97%	96%/49%	90%/87%	91%/94%	97%/93%
state	23%/23%	93%/95%	97%/93%	22%/22%	93%/95%	97%/93%
title	9%/10%	33%/18%	31%/43%	3%/3%	38%/33%	35%/35%
salary	55%/55%	56%/70%	64%/54%	46%/48%	62%/72%	68%/72%

Table 4: Comparison between single loop information extraction (only word-based classification using Ripper) and the double classification approach. For the latter approach we report results from using the best relevant sentence and using all relevant sentences for the word-level classification. We report (precision/recall).

	AO			OBD		
	recall	precision	F_1	recall	precision	F_1
exact	70%	54%	61%	70%	59%	64%
contains(2)	73%	55%	63%	72%	60%	65%
overlaps(2,1)	77%	59%	67%	77%	65%	70%

Table 5: Influence of the operator used to evaluate on the overall score, minus `language` and `area`. All relevant sentences were used, feature vector is default with class-before.