
Automata learning for numerical entities extraction from OCR output

Christopher Kermorvant
Abdelhalim Rafrafi

A2iA, 40 bis rue Fabert, 75007 Paris, France

CK@A2IA.COM
AR@A2IA.COM

Abstract

In this article, we study the use of a grammatical inference algorithm (RPNI) to infer information extraction models from positive and negative examples. We compare classical regular expressions and inferred automata in a real information extraction task defined on digitalized documents processed by OCR. We show that the automata inferred from ground-truth values and the output of our OCR outperform an extraction model based on regular expressions.

1. Information extraction in Document Data Capture Industry

Information extraction is a task which belongs to the broader field of text mining. Whereas the latter consists in extracting subtle or at least non trivial knowledge from text, information extraction mostly consists in extracting entities from text. From the pioneering MUC conferences in the nineties (Hirschman, 1998) to the latest Pascal challenges (Ireson et al., 2005), annotated corpora have been gathered, metrics have been defined and several systems based on machine learning algorithms have been compared.

In most of the benchmarks, the extraction tasks are defined on text from email, web pages or electronic news. Some of the information extraction systems have been tested on text produced by a speech recognizer (Favre et al., 2005; Horlock & King, 2003) and very few on output produced by writing recognition systems (Miller et al., 2000; Chatelain et al., 2006). In the case of information extraction from text produced by speech or writing recognizer, not only the punctuation and the capitalization are removed but recognition

errors (insertion, deletion, substitution) are introduced at word level and also at character level in the case of writing recognizers. The specificities of recognized text make information extraction more difficult and simple tasks like date extraction become not trivial.

Administrations and large companies must with deal a growing volume of documents and despite the increasing use of electronic documents, the volume of the paper documents that are exchanged daily is still very large. The Document Data Capture Industry provides solutions (scanner, software) to process electronic and paper documents and aims at reducing the associated cost and processing time. Paper documents are usually scanned before being included in the electronic document workflow so that only an image of these documents is exchanged between the different agents involved in the document process. Most of the software solutions include recognition engines for document routing, classification or information extraction.

Automatic information extraction is still in its infancy in the industry of digital document management, so that complex tasks such as text understanding or summarizing are not yet considered by most customers or end-users. In the context of form processing or incoming mail processing, companies generally define information extraction tasks as the following, in order of increasing difficulty:

- extraction of an identification number such as social security number, customer phone number, account numbers;
- extraction of a simple alpha-numerical value such as a date, the amount of an invoice or transfer, a name in a list of possible names.
- extraction of a complex alpha-numerical identifier such as the complete bank account number (bank number, office number, account number and key number);

- extraction of a complex entity composed of several alpha-numerical identifier or values with specific relations between them, such as a complete transfer order, or a complete address.

In the industry, information extraction tasks largely correspond to named entities extraction as defined in academic research laboratories but with more emphasis put on numerical entities extraction. In academic research laboratories, the extraction of numerical entities is often considered as relatively simple and therefore has received little attention. However, in the context of extraction from the output of a writing recognizer, the difficulty of numerical entities extraction largely increases and requires the use of more sophisticated models.

2. Information extraction with regular expressions

Regular expressions are a simple and popular model for information extraction from documents processed by an automatic recognition engine. They provide a simple way for end-users to define the format of the information that they want to extract that can be used directly in the extraction engine. Most of the products available on the market for information extraction in recognized documents embed regular expressions as an interface for the definition of information extraction tasks.

However, regular expressions have several limitations. First, the definition by the end-user of the format of the information to extract is rarely exhaustive: even on simple entities like dates, several formats are often encountered in documents (for example '02/03/2005', 'March, 2 2005', '2/3/05'). The end-user can never give a regular expression which encompasses all the forms of the information to extract.

Second, the regular expression is adapted to electronic text but not to the noisy text produced by a recognizer. For example, it is usually not possible to discriminate the letters O and l from the numbers 0 and 1 on printed documents. Moreover, when document images are noisy or when documents are scanned at low resolution as is the case for faxes, more confusion occurs, between 5 and S, 6 and G, etc. The regular expression defined by the end-use does not take into account all these possible confusions.

The problem of adapting regular expressions with examples can be tackled by using regular inference algorithms. Amongst all the algorithms proposed for regular grammatical inference, RPNI (Oncina & García,

1992) is still the most popular. This algorithm infers from positive and negative examples a regular language as a finite state automata. This automata can be easily converted into a regular expression and plugged into an information extraction engine based on regular expression.

Grammatical inference has already been applied to infer models for information extraction, for example to adapt the structure of an extraction model based on HMM (Freitag & McCallum, 2000), or to learn the context of named entities (Talukdar et al., 2006). In all these works, grammatical inference is used to infer the structure of a phrase or a sub-phrase containing an entity to extract. To our knowledge, grammatical inference has never been used to infer the structure of the entity itself, as proposed in this work.

In this article, we study the use of RPNI to infer information extraction models from positive and negative examples. We test several models of naive regular expressions and of automata inferred on positive and negative examples extracted from both ground-truth values of entities and document recognition results. We show that learning information extraction models from examples significantly improves the performance of the A2iA extraction engine on a real information extraction task.

The article is structured as followed: we first present the architecture and the principles of A2iA DocumentReaderTM, A2iA engine for processing all types of documents. We compare several information extraction models based on regular expression or regular automata on a real information extraction task from faxed documents. Finally, we propose some possible developments in the inference algorithms and in the design of a end-user interface to learn information extraction models based on examples.

3. A2iA Information Extraction System

A2iA DocumentReaderTM is a comprehensive engine for processing of all types of documents from purely printed structured forms to free handwritten letters. The engine offers full transcription of printed, hand-printed and cursive writing. It also provides automatic information extraction and document classification based both on layout and content.

The architecture of the engine for the complete processing of a document can be decomposed in 4 parts:

1. image binarization and pre-processing: resolution correction, denoising, orientation correction
2. document layout analysis: identification of the

document zones corresponding to text, geometrical elements (lines, table) and graphical elements (logo)

3. recognition of any kind of writing: printed (OCR), hand-printed (separated numbers and capital letters) and cursive hand-writing
4. document classification and information extraction based on the previous stages and according to the end-user configuration

The first stage consists of several image processing procedures to convert gray scale images to binary images, to normalize the image resolution (fax images are often in 200*100 resolution), to remove salt and pepper noise, to find the main orientation of the document (the orientation of most of the text).

The second stage consists in computing skeleton arcs and clustering them bottom-up into coherent objects according to local and *a priori* statistics and arc morphology. These objects are then assigned to a category amongst geometrical objects, machine-printed text, hand-printed text, handwritten, symbol, and texture. Finally, each text zone is processed by a specialized recognizer according to its writing type (printed, hand-printed, cursive).

The printed and hand-printed recognizers are both based on the same principles (see (Gorsky, 2007) for details). First text lines are segmented into potential characters images which are processed by a neural network-based OCR. Once characters are recognized, words candidates are stochastically generated by combining character classes with their probabilities, probability of character segmentation options and word segmentation options. A list of word candidates is generated and may be filtered by a dictionary of possible words if available.

Once the full transcription of the page has been done, each recognized text zone is matched with a regular expression defining the entities to extract. If the text zone matches, the exact zone of the matching entity is extracted at character level. This zone is then processed by a specialized recognizer depending on the type of the entity (date, SSN, numerical, etc.).

In this work, we propose to use grammatical inference to learn the model which detects entities in the text produced by the full page transcription.

4. Comparison of information extraction models

4.1. The information extraction task

The information extraction task was provided by one of our customers. This customer is a service bureau company which provides information processing services and softwares for financial companies. This service bureau processes all the documents sent to its clients. These documents are faxes or digital images of complete mails containing official structured forms, investment forms, bank cheques, envelops, address change forms, death certificates, etc. The document flow is largely heterogeneous so that it is not possible to extract all the interesting information at fixed position. This information must be localized and extracted in the complete document and possibly on several pages.

On these documents entities to extract are social security numbers (SSN), dates, account numbers, phone numbers and proper names. We have manually annotated 3193 documents with the location, the type (date, phone number, SSN, account number) and the ground-truth value of all the 14 713 entities found on the images: 7930 dates, 1698 phone numbers, 1545 SSN, 3540 account numbers.

From these locations and true values, we have generated 2 different positive and negative set of examples for each type of entities as presented on Figure 1. The first set of examples was composed of OCR outputs (recognition examples) and the second set of example was composed of true values.

The first set was generated as follows : from the location of the entities, we have extracted snippets and we have typed them as positive or negative examples, depending on the model we wanted to learn, as presented on Table 1. We processed each snippet with a generic OCR (not specialized for numerical entities). The hand-printed and cursive zones were not processed by the OCR. At the end of the recognition process, we had 8202 entities : 4371 dates, 1114 phone numbers, 594 SSN and 2123 account numbers. The difference between the number of annotated entities (14713) and the number recognized entities (8202) is due either to the writing style (handwritten entities are not recognized by the OCR) or to very noisy images on which the OCR does not give any recognition result.

The set of true value examples was simply generated by selecting true values according to their type and to the model to learn, as presented on Table 1.

It is important to note that we focused on the ex-

Figure 1. Generation process of positive and negative examples for training and testing the models

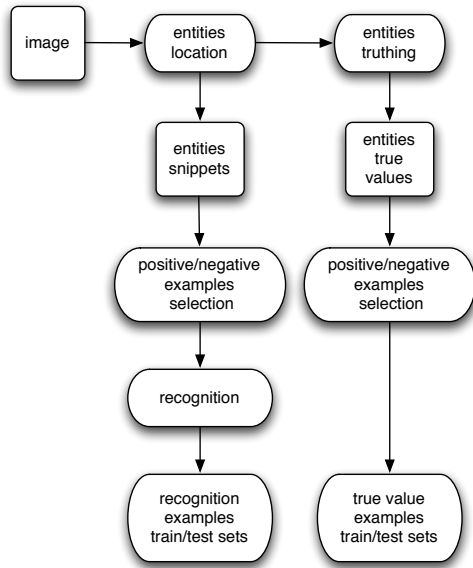


Table 1. Positive and negative examples for each extraction model.

EXTRACTION MODEL	POSITIVE EXAMPLES	NEGATIVE EXAMPLES
DATES	DATES	SSN, PHONE NUMBERS, ACCOUNT NUMBERS, NAMES
SSN	SSN	DATES, PHONE NUMBERS, ACCOUNT NUMBERS, NAMES
PHONE NUMBERS	PHONE NUMBERS	DATES, SSN, ACCOUNT NUMBERS, NAMES

traction of numerical entities only. In the extraction task on this database, most of the confusions occur between different types of numerical entities (SSN, account number, phone number, date) and most of them must be extracted and correctly typed.

4.2. Description of the models and evaluation metrics

We here compare five different extraction models based on regular expressions or finite state automata. These models are :

- **A priori** : this model is the *naive* regular expression that a end-user would define to extract the information he needs. Regular expressions for

 Table 2. Regular expressions for *a priori* models

MODEL	A PRIORI REGULAR EXPRESSION
DATE	<code>\d\d[-/]\d\d[-/](\d\d\d\d\d\d)</code>
SSN	<code>\d\d\d[-]\d\d[-]\d\d\d</code>
PHONE NUMBER	<code>\(? \d\d\d(\[-])? \d\d\d[-]? \d\d\d\d</code>

Table 3. Regular expressions for hand-made models (using python syntax)

MODEL	HAND-MADE REGULAR EXPRESSION
DATE	<code>([\dZOISG]{1,2}[-/]*){2}[\dZOISG]{2,4}</code>
SSN	<code>[\dZOISG]{3}[-/] * [\dZOISG]{2}[-/] * [\dZOISG]{4}</code>
PHONE NUMBER	<code>\(?[\dZOISG]{3}(\[-] * [\dZOISG]{2}[-/] * [\dZOISG]{4}</code>

these models are given in Table 2. Both Table 2 and Table 3 use the python syntax¹ to define regular expressions : in particular `\d` matches any decimal digit, and each capital letter (Z,O,I,etc) matches itself.

- **Hand-made** : this model is the *a priori* model which has been adapted by hand by a recognition expert to take into account the variability in the form of the information and the recognition errors to a certain extent. Regular expressions for these models are given in Table 3;
- **RPNI-T** : this model is an automaton inferred using RPNI from positive and negative examples obtained by human truthing of the information to extract;
- **RPNI-R** : this model is an automaton inferred using RPNI from positive and negative examples obtained by automatic recognition of the documents (OCR output)
- **RPNI-TR** : this model is an automaton inferred using RPNI from positive and negative examples obtained by mixing human truthing and automatic recognition of the documents.

RPNI (Oncina & García, 1992) is a state merging algorithm for finite state automata inferring from posi-

¹<http://docs.python.org/lib/re-syntax.html>

tive and negative examples. The starting point of the algorithm is a tree shape automata called prefix tree automata (PTA) which exactly accepts all the positive examples and rejects all the negatives examples. The main loop of the algorithm consists in running through the PTA and trying to merge each couple of states : if the merge of 2 states leads to a new automata \mathcal{A} which does not accept negative examples, \mathcal{A} replaces the current automata, after some more fusions to keep \mathcal{A} deterministic. When no more state fusion is possible, the algorithm stops and return the current automata.

The complete database was randomly split into a training part (80%) and a testing part (20%). The automata were inferred by RPNI on the training set. All the models were tested on the test set using the classical precision/recall/f-measures measures. For example, in the case of a date extraction model, each entities in the test set (date, SSN, account numbers, phone numbers) was presented to the extraction model, which decided whether or not it was a date. If the entity was really a date (resp. not a date) and that the models decided that it was a date (resp. not a date) the entity was correctly detected. In other cases, the entity was not correctly detected. For a given type E of entities, evaluation measures were computed as follows:

- precision : $\frac{\# \text{ entities detected E and really E}}{\# \text{ entities detected E}}$
- recall : $\frac{\# \text{ entities detected E and really E}}{\# \text{ entities E in the test set}}$
- f-measure : $\frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$

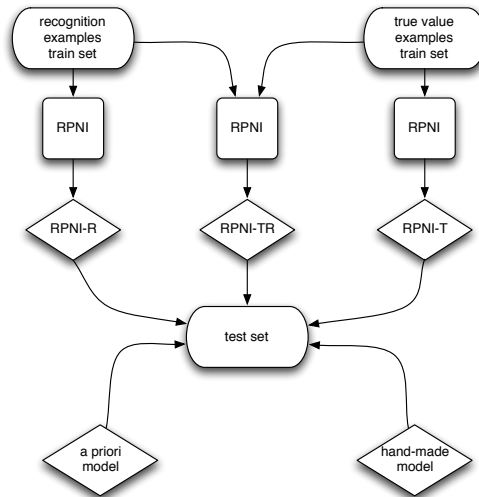
Figure 2 summarizes the training and testing process and the 5 models.

It is important to note that our goal was to learn an extraction model, which is used to locate entities on the document. This entities location is followed, in the complete extraction process, by a second recognition performed by a specialized recognizer. When working on recognition output, we never compared the value of the extracted entities to the true value, since this comparison would sum two kind of errors : extraction errors and recognition errors.

4.3. Extraction results

Table 4 presents the extraction results of the five models on the recognition test set (OCR output) for the 3 types of numerical entities, phone numbers (TEL), social security number (SSN), and dates (DAT).

Figure 2. Training and testing process for the five tested models.



The *a priori model* has a very high precision on SSN and dates, which is due to its high specificity : SSN and dates containing only numbers and following the regular expression are detected. On phone numbers, the *a priori* regular expression gives poor results both in precision and recall : this is due to the fact that phone numbers have weak structural constraints. They are mainly a sequence of numbers and thus can be easily confused with account numbers.

The hand-made model does not significantly improve the results compared to the *a priori* model. Taking into account usual recognition errors improves the recall on phone numbers, SSN and dates but reduces the precision, leading to a f-measure similar to that of the *a priori* model (except for SSN).

Overall, using RPNI to learn an extraction model increases the recall and reduces the precision but leads to models with better f-measures. For phone numbers, the recall increases from 0.3 with regular expression models to 0.6 with models learnt with RPNI. For dates, the recall increases from 0.66 with regular expression models to 0.87 with models learnt with RPNI. The precision slightly decreases for dates and SSN and but increases for phone numbers.

Amongst all the models inferred with RPNI, the best models use examples from both true-values and recognition outputs. For dates, the automaton inferred on true-values has logically better precision and lower recall than the automaton inferred on recognized strings. The combination of the two sources of examples leads to an automata with better f-measure than the two

Table 4. Comparison of the five models for the extraction of phone numbers (TEL), social security numbers (SSN), and dates (DAT) on the recognition test set (OCR output), according to their precision (P), recall (R) and f-measure (F).

MODEL	ENTITY	P	R	F
A PRIORI	TEL	0.27	0.32	0.30
HAND-MADE	TEL	0.27	0.33	0.30
RPNI-T	TEL	0.85	0.45	0.59
RPNI-R	TEL	0.56	0.40	0.47
RPNI-TR	TEL	0.79	0.48	0.60
A PRIORI	SSN	0.98	0.53	0.69
HAND-MADE	SSN	0.80	0.70	0.75
RPNI-T	SSN	0.76	0.64	0.70
RPNI-R	SSN	0.33	0.27	0.30
RPNI-TR	SSN	0.76	0.64	0.70
A PRIORI	DAT	1.00	0.49	0.66
HAND-MADE	DAT	0.80	0.52	0.63
RPNI-T	DAT	0.96	0.77	0.85
RPNI-R	DAT	0.78	0.84	0.81
RPNI-TR	DAT	0.88	0.85	0.87

automata separately. In the case of SSN detection, the inferred models are not significantly better than the a priori or hand-made models. This might be due to the limited number of examples compared to the variability of the SSN.

Table 5 shows the size of the automata (number of states and number of transitions) inferred by RPNI-TR for different size of the training set and for the three entities (date, SSN, phone number). The number of positive and negative example in the training set is equal if possible, or limited by the maximum number of positive examples available. Note that the size of the inferred automata does not seem to reach a stable value as the number of examples increases, which means that more examples would still improve the automata.

Table 6 presents the extraction results of the five models on the true-value test set. By comparison with Table 4, one can evaluate the impact of the recognition process on the entity extraction. The recognition has an large impact on the recall score for all types entity. For the *a priori model*, the recall drops from 0.73 to 0.32 for telephone numbers, from 0.71 to 0.53 for SSN and from 0.73 to 0.49 for date due to recognition error. Recognition errors also induce confusions between entities, mostly for phone numbers: the precision of the *a priori model* drops from 0.42 to 0.27.

Note that even on clean data, for SSN and phone num-

Table 5. Size of the inferred automata (number of states and number of transitions) for different sizes of training set (50% of positive examples and 50% of negative examples) and for Date, SSN and phone numbers.

ENTITY	TRAINING SIZE	NB STATES	NB TRANS.
DAT	200	6	105
DAT	400	8	132
DAT	1000	15	217
DAT	1400	15	293
DAT	2000	22	353
DAT	5000	33	592
DAT	10000	52	901
DAT	18829	65	1197
DAT	26219	80	1434
SSN	200	9	85
SSN	400	10	112
SSN	1000	21	322
SSN	1400	29	401
SSN	2000	33	498
SSN	4221	58	814
SSN	6721	73	918
SSN	11721	84	1041
SSN	26292	101	1216
TEL	200	8	105
TEL	400	11	152
TEL	1000	21	312
TEL	1400	27	400
TEL	2000	31	473
TEL	4695	59	935
TEL	7195	73	1061
TEL	11415	86	1187
TEL	26275	99	1294

ber extraction, the automata inferred by RPNI underperform the *a priori* or hand-made regular expressions. We think that this is due to the small number of positive examples available for these two entities.

Finally, Figures 3, 4 and 5 present several learning curves for the RPNI-TR model. It shows the performances of this model for date extraction on the recognition test set (OCR output) for an increasing number of positive and negative examples. As expected, increasing the number of negative examples improves the precision (Figure 3). On the other side, increasing the number of positive examples improves the recall (Figure 4). It is also worth noting that the algorithms needs an large number of positive and negative examples to reach its best results (Figure 5): with 1000 positive and 1000 negative examples, the f-measure is at 0.76, whereas it reaches 0.87 with 9917 positive and 16302 negative examples.

Table 6. Comparison of the five models for the extraction of phone numbers (TEL), social security numbers (SSN), and dates (DAT) on the true-value test set, according to their precision (P), recall (R) and f-measure (F).

MODEL	ENTITY	P	R	F
A PRIORI	TEL	0.42	0.73	0.54
HAND-MADE	TEL	0.42	0.73	0.54
RPNI-T	TEL	0.46	0.37	0.41
RPNI-R	TEL	0.46	0.36	0.41
RPNI-TR	TEL	0.53	0.41	0.47
A PRIORI	SSN	0.96	0.71	0.82
HAND-MADE	SSN	0.86	0.90	0.88
RPNI-T	SSN	0.53	0.54	0.53
RPNI-R	SSN	0.54	0.40	0.46
RPNI-TR	SSN	0.61	0.60	0.61
A PRIORI	DAT	1.00	0.73	0.84
HAND-MADE	DAT	0.88	0.74	0.81
RPNI-T	DAT	0.97	0.94	0.95
RPNI-R	DAT	0.89	0.70	0.78
RPNI-TR	DAT	0.95	0.94	0.95

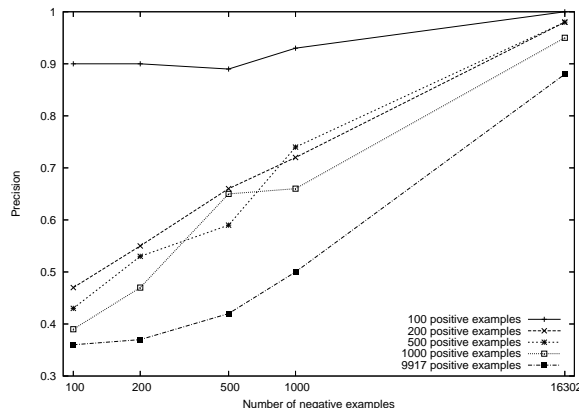
5. Conclusion and perspectives

We have presented an application of the regular inference algorithm RPNI to the inference of numerical entities extraction models. The models inferred with RPNI on positive and negative examples extracted from both true-values and recognition outputs outperform the usual models based on hand-made regular expressions. We still have to validate this approach in a complete extraction process, by using our model to locate entities and run a specific recognizer on the detected entities. In this complete process, the evaluation of the extraction also takes into account the recognition errors.

The perspectives of improvement of our extraction models based on regular inference are numerous, since we have tested only one inference algorithm and we did not include any probabilities in our models.

First we can evaluate the use of non-recursive automata as extraction models, as proposed by (Talukdar et al., 2006) with k-testable automata. Second, we can include probabilistic information in our models by inferring stochastic automata (Carrasco & Oncina, 1994) in order to have a matching score associated to the extraction. We can also weight our training examples according to their recognition probabilities in order to decrease the influence of badly recognized entities in the inference of the automata. The extraction models must also be trained in context, in order to take advantage of the possible cues in the neighborhood of the entities (e.g. keywords like "SSN", "date:", etc.)

Figure 3. Learning curves : precision of the RPNI-TR model for date extraction on the recognition test set (OCR output) with respect to the number of negative examples and for different numbers of positive examples in the training set.



has it is usually done in entities extraction systems.

Finally we can include a priori information, like the regular expression provided by the end-user in the inference algorithm (Kermorvant & de la Higuera, 2002). This is a way to reduce the number of positive and negative examples needed. This point is important for industrial products since the examples must be provided by the end-user and that he is usually not ready to spend a lot of time locating and truthing entities on his documents.

References

- Carrasco, R. C., & Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. *Proc. Int. Coll. on Grammatical Inference* (pp. 139–152). Springer.
- Chatelain, C., Heutte, L., & Paquet, T. (2006). Discrimination between digits and outliers in handwritten documents applied to the extraction of numerical fields. *International Workshop on Frontiers in Handwriting Recognition, La Baule, France* (pp. 475–480).
- Favre, B., Bchet, F., & Nocera, P. (2005). Robust named entity extraction from large spoken archives. *Proc. Conf. on Empirical Methods in Natural Language Processing*.

Figure 4. Learning curves : recall of the RPNI-TR model for date extraction on the recognition test set (OCR output) with respect to the number of positive examples and for different numbers of negative examples in the training set.

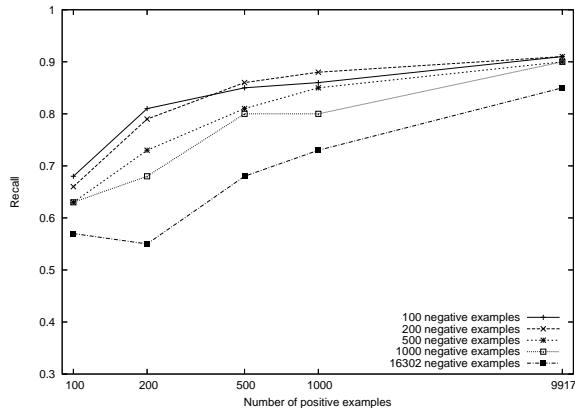
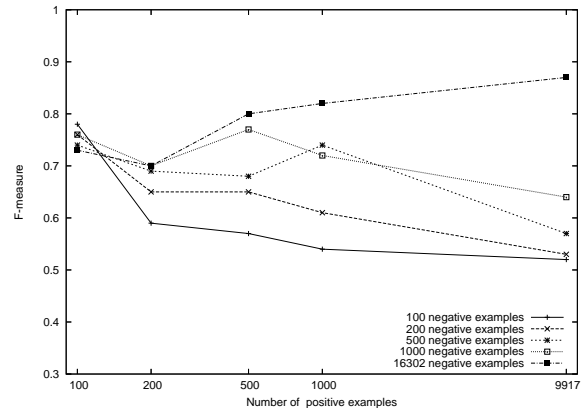


Figure 5. Learning curves : f-measure of the RPNI-TR model for date extraction on the recognition test set (OCR output) with respect to the number of positive examples and for different numbers of negative examples in the training set.



Freitag, D., & McCallum, A. (2000). Information extraction with HMM structures learned by stochastic optimization. *Proceedings of the Eighteenth Conference on Artificial Intelligence*.

Gorsky, N. (2007). *Digital document processing*, chapter Bank Cheque Data Mining: Integrated Cheque Recognition Technologies, 437–458. Springer London.

Hirschman, L. (1998). The evolution of evaluation: Lessons from the message understanding conferences. *Computer Speech and Language*, 12, 281–305.

Horlock, J., & King, S. (2003). Discriminative methods for improving named entity extraction on speech data. *Proc. European Conf. on Speech Communication and Technology* (pp. 2765–2768).

Ireson, N., Ciravegna, F., Califf, M. E., Freitag, D., Kushmerick, N., & Lavelli, A. (2005). Evaluating machine learning for information extraction. *Proc. Int. Conf. on Machine Learning*.

Kermorvant, C., & de la Higuera, C. (2002). Learning languages with help. *Proc. Int. Coll. on Grammatical Inference* (pp. 161–173). Springer.

Miller, D., Boisen, S., Schwartz, R., Stone, R., & Weischedel, R. (2000). Named entity extraction from noisy input: Speech and ocr. *Proc. of the North*

American chapter of the Association for Computational Linguistics annual meeting (pp. 316–324).

Oncina, J., & García, P. (1992). Identifying regular languages in polynomial time. In H. Bunke (Ed.), *Advances in structural and syntactic pattern recognition: Proc. of the international workshop*, 99–108. World Scientific.

Talukdar, P. P., Brants, T., Liberman, M., & Pereira, F. (2006). A context pattern induction method for named entity extraction. *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)* (pp. 141–148).