A Visual and Results-Driven Rules Composition Approach for Better Information Extraction

Wassim El-Kass*. Stéphane Gagnon.**
Michal Iglewski***

** Computer and Engineering Department, University of Quebec in Outaouais (UQO)*
Gatineau, Canada, (e-mail: Wassim.El-Kass@uqo.ca)}
*** Department of Administrative Sciences, University of Quebec in Outaouais (UQO)*
Gatineau, Canada, (e-mail: Stephane.Gagnon@uqo.ca)}
**** Computer and Engineering Department, University of Quebec in Outaouais (UQO)*
Gatineau, Canada, (e-mail: Michal.Iglewski@uqo.ca)}

Abstract: We present a highly visual process for creating and combining elementary information extraction rules, based on their results, in order to find the rules combination that produces the most accurate information extraction results. A rule's accuracy is determined by its F-Score which is the harmonic mean of the precision and the recall of that rule. Rules are combined using logical OR and AND operators. Running a few hundreds rules combinations over a corpus, in order to determine their accuracies, can take days. Using our approach, millions of rules combinations can be tested and their accuracies (F-Score) can be calculated in few seconds. A prototype was created to demonstrate the effectiveness of our approach.

Keywords: Information Extraction, Rules Composition, Automation, Visualization.

1. INTRODUCTION

Information Extraction (IE) and Information Retrieval (IR) are getting increased interest due to the rise of "Big Data" and the unprecedented amount of unstructured and semi-structured data generated by computers and mobile systems. Easier and more accurate and efficient IE and IR systems are needed to quickly identify relevant and actionable information which is crucial for search engines and any facts-based decision support system**.**

IE and IR can be rule-based or based on statistical machine learning (ML) algorithms. A recent study done by IBM researchers showed that academic research was mainly focused on ML IE while almost totally ignoring rule-based IE in the last decade (Chiticariu, Li, & Reiss, 2013). The same study discusses the importance of rule-based IE and the need for advancing the state-of-the-art in rule-based IE systems.

A big challenge in rule-based IE is to create and maintain a comprehensive set of annotation rules and finding the rules combination that produces the most accurate IE results. In this paper, we present a process and an algorithm that generates the most accurate rules combination based on the results of individual rules over a training set. We developed a prototype to demonstrate the simplicity and effectiveness of our approach.

2. MEASURING RULES ACCURACIES

We use the F-Score to measure the accuracy of rules. The F-Score is the harmonic mean of the precision and the recall. The precision of a rule is the ratio of the number of relevant results matched by the rule (True Positives or TP) to the total number of results matched by the rule including the irrelevant results (False Positives or FP) as in (1). A rule's recall is the ratio of True Positives (TP) to the number of all relevant results including those not matched by the rule (False Negatives or FN) as in (2). Equation (3) shows the formula for calculating the F-Score of a rule using its precision and recall.

$$precision = \frac{TP}{TP + FP} \qquad (1)$$

$$recall = \frac{TP}{TP + FN} \qquad (2)$$

$$F - Score = \frac{2 \times precision \times recall}{precision + recall} \qquad (3)$$

The precision is directly proportional with TP and inversely proportional with FP. The recall is directly proportional with TP and inversely proportional with FN. Therefore, the focus should be on increasing the TP and reducing the FP when only the precision matters. To get a better recall, the TP should be increased while decreasing the FN.

The Receiver Operating Characteristic (ROC) is often recommended along with the F-Score. It is widely applied in evaluating classification algorithms (Bradley, 1997). It is relevant to our types of rules, with conditional and statistical matching (first-order logic) and annotation graphs (second-order logic) (Michaelis & Mönnich, 2007). Our next release will apply the methods presented here to ROC curves as well.

3. VISUALIZATION

Visualization is employed in various domains to reduce complexity. Rule-based IE systems are usually criticised for the complex and tedious manual labour required for building and maintaining the rules and for finding the rules combination that produces the most accurate results. We therefore suggest the use of three visualization types namely data visualization, rules visualization, and rules results visualization in addition to automatic rules composition when preparing rules for rule-based IE systems. The following sections give more details about these concepts.

3.1 Data Visualization

Data visualization is a way to present the data or part of it in a visual way; often using graphical shapes such as charts, gauges, geographical maps, heat maps, histograms, pies, and so on; to highlight patterns, trends and correlations that are hard to notice by looking directly at the data. Data visualization has been used to deal with the data volume and complexity in data mining and IE systems (D. A. Keim, 2002). Data visualization also plays a key role in analysing big data as it gives a quick insight into the vast amounts of data and helps driving complex analysis (Fayyad, Wierse, & Grinstein, 2002; D. Keim, Qu, & Ma, 2014). Many tools, including Tableau and QlikView, exist today for visualizing structured as well as unstructured data. Using these tools over a training set can help understanding the data and finding common patterns, trends, and correlations. This can be crucial for building the initial set of elementary rules in a rule-based IE system.

3.2 Rules Visualization

Creating and maintaining rules in rule-based IE systems is a tedious task and is usually done by highly technical users especially when extracting information from unstructured data. This is mainly because most rules languages and editors are based on regular expression grammar or programming libraries that are complex even for advanced software developers (Reiss, Raghavan, Krishnamurthy, Zhu, & Vaithyanathan, 2008). Visual rules creation and manipulation enables non-technical domain experts to create and maintain IE rules for their domains.

UIMA Ruta (formally TextMarker) and JAPE are amongst the best rules languages for IE from unstructured text with UIMA Ruta being more on the rise (Cunningham, Maynard, & Tablan, 1999; Kluegl, Toepfer, Beck, Fette, & Puppe, 2014; Toepfer, 2014). Both languages offer a rich set of operations for matching patterns in unstructured text but have a complex syntax and require advanced programming skills.

3.2.1 ARDAKE

We developed ARDAKE (Fig. 1) as a visual IE rules editor to overcome the complexity of rules languages and to enable non-technical domain experts to easily create and maintain complex IE rules. Rules in ARDAKE are created using simple mouse drag/drop of various built-in or user-defined artefacts. Built-in artefacts include simple patterns like 'white space', 'number', 'word', and 'sentence', and more advanced patterns such as html and xml elements and concepts defined in OWL ontologies. Other ARDAKE artefacts include conditions to check if a pattern contains, begins with, or ends with another pattern, and many more linguistic and semantic conditions. The most important action artefact in ARDAKE is the 'Mark as' action that marks a specific portion of the text as a pattern of a given type but ARDAKE supports a number of other actions for manipulating patterns.

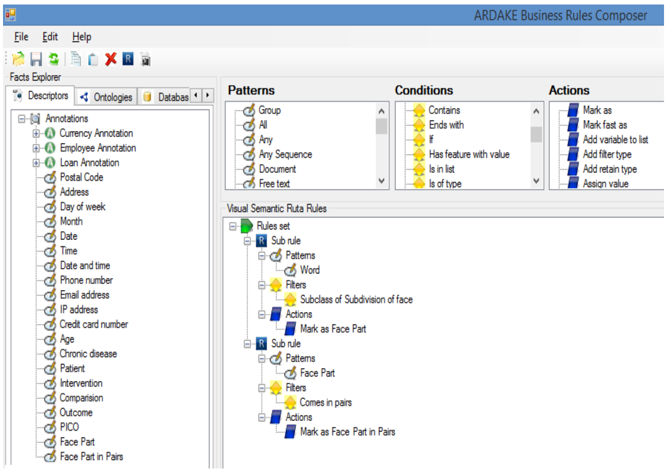


Fig. 1. Our visual IE rules editor ARDAKE.

An ARDAKE rule has three parts; a patterns part where users can specify a pattern or a sequence of mixed mandatory and optional patterns to match in a corpus; an optional conditions part where multiple conditions can be combined using logical operators to filter annotations matching the pattern(s) under the patterns part; and an actions part to apply any number of actions on annotations resulting from the first two parts of the rule.

Fig 2. Shows a simple visual ARDAKE rule to mark any sentence that ends with a question mark as an interrogative sentence. The first line is a textual description of what the rule does. The patterns part of this rule contains one pattern "Sentence" to indicate that this rules is only applied on sentences. The filters part contains one condition "Ends with ?" to filter out sentences that do not end with a question mark. The only action in the actions part of the rule is to mark (i.e. annotate) any sentence that ends with a question mark as an interrogative sentence.

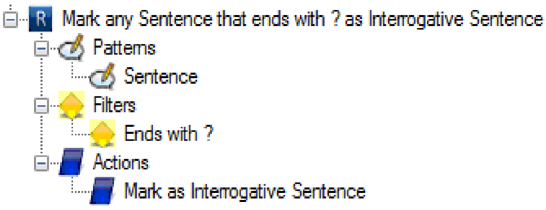

Mark any Sentence that ends with ? as Interrogative Sentence
Patterns
Sentence
Filters
Ends with ?
Actions
Mark as Interrogative Sentence

Fig. 2. ARDAKE rule for matching interrogative sentences.

ARDAKE automatically adds user-defined patterns, like Interrogative Sentence in Fig.2, to the list of existing patterns so they can be used when defining new rules. This allows ARDAKE users to create and rules libraries for different domains like health, insurance, e-commerce and so on.

The following code snippet shows the UIMA Ruta code that ARDAKE generated for the rule in Fig.2.

```
DECLARE Sentence;
PERIOD # { -> MARK(Sentence)}
PERIOD;

DECLARE QuestionMark;
"?" { -> MARK(QuestionMark)};

DECLARE InterrogativeSentence;
Sentence {ENDSWITH(QuestionMark) ->
MARK(InterrogativeSentence)};
```

ARDAKE can be used as a visual UIMA Ruta rules editor since it supports all Ruta constructs in an intuitive visual way that makes it easy for non-technical domain experts to get the full power of the Ruta rules language without having to worry about its complex syntax. In fact, ARDAKE extends Ruta by adding semantic patterns for matching complex patterns defined in OWL ontologies; semantic conditions such as SubClassOf to test if a pattern is a direct or indirect subclass of another concept in an ontology; and semantic actions like “AddToOntology” to add a matched concept to a given ontology.

ARDAKE itself can be extended by developers who wish to add more patterns, conditions, or actions. This can be done by implementing well defined ARDAKE interfaces and adding the implementation as an ARDAKE plugin.

3.3 Rules Results Visualization

A rule’s accuracy is determined by the results it produces when executed over a training set. To measure the F-Score of a rule, its TP, FP, and FN should first be calculated. These measurements along with the True Negatives (TN), i.e. results successfully identified as non-relevant by the rule, can be analysed to figure out where and why a rule is poorly performing. For example, a rule that generates lots of FP could give an indication that the rule’s condition is too loose and needs to be more restrictive. Similarly, a rule that produces lots of FN may require to match on more patterns and/or have its condition(s) relaxed in a way to filter non-relevant results while keeping the relevant ones.

Analysing the results of different rules also helps determining what logical operators to use for combining specific rules in order to obtain compound rules with higher F-Scores. Rules that have lots of TP in common and few common FP should be combined using the logical AND operator while those that share more FP and less TP are better combined using the logical OR operator.

Visualizing “rules results” gives an immediate insight into their quality by graphically showing the proportion of each measurement. This could be useful to determine which rules to combine using what logical operator in order to obtain more accurate results.

Fig. 3 and Fig. 4 show the visual representations of the results of a (high precision, low-recall) rule and a (low precision, high recall) rule respectively. The portion between the solid lines (relevant portion) in the pie chart represents all relevant results for a specific query. The portion between the dashed lines (rule results portion) represents the results returned by a rule. The intersection between the relevant portion and the rule results portion is the TP portion of the rule. FP is the sub portion of the rule results portion that is not part of the relevant portion while FN is the sub portion of the relevant portion that is not part of rule results portion. Finally, the TN portion is the portion that is outside the relevant and the rule results portions.

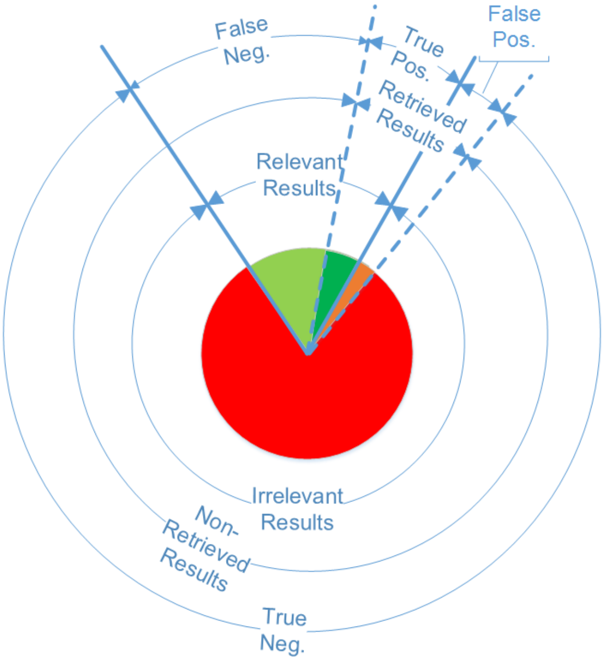


Fig. 3. Rule 1: High Precision-Low Recall rule results

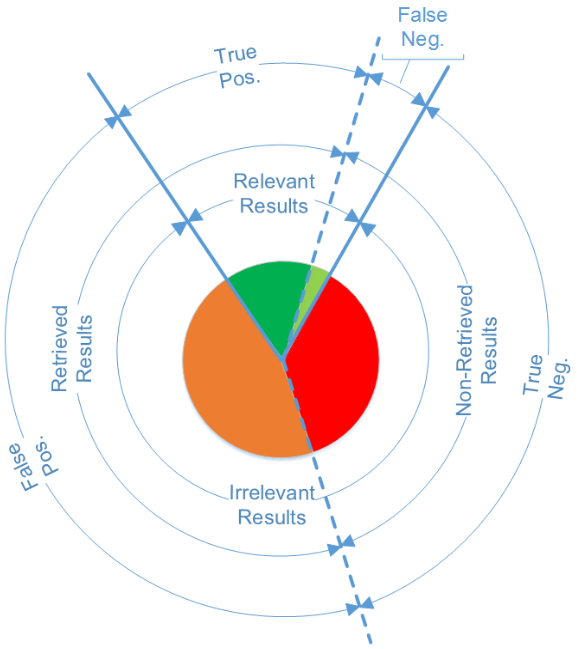


Fig. 4. Rule 2: Low Precision-High Recall rule results

Combining the above rules using the logic AND operator is likely to produce a rule with a higher precision and a lower recall than both rules as shown in Fig. 5. Combining the same rules using the logic OR operator is likely to produce a rule with a lower precision and a higher recall than both rules as shown in Fig. 6.

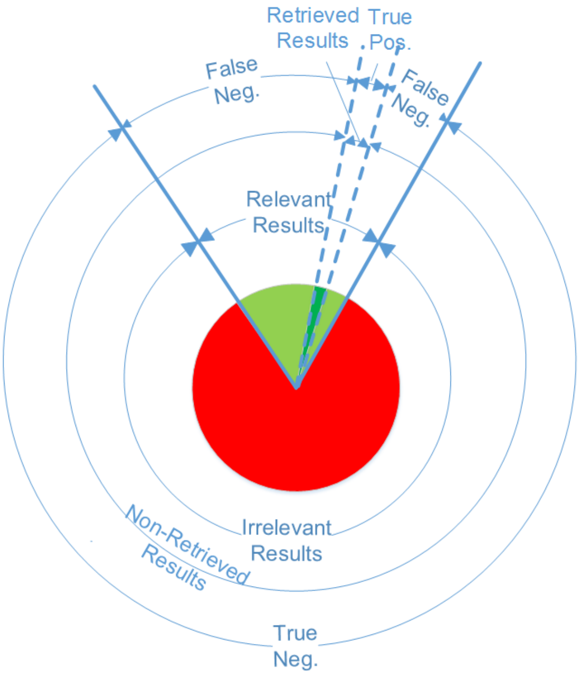


Fig. 5. Results of Rule1 AND Rule2

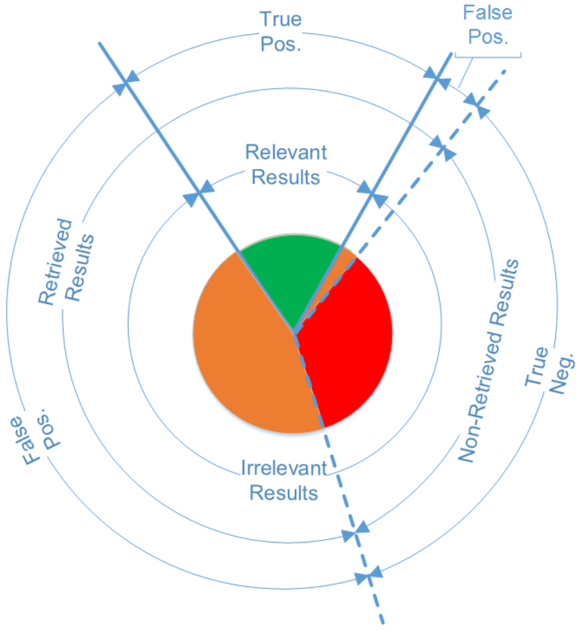


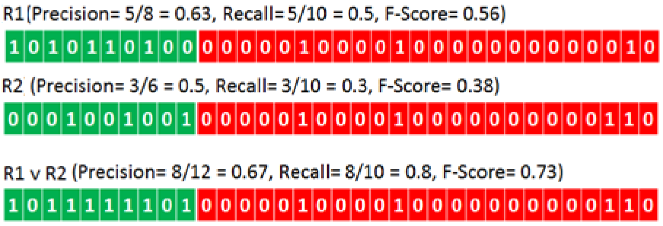
Fig. 6. Results of Rule1 OR Rule2

Manual rules composition can be done on a very small rules set. Automatic rules composition should be considered when dealing with more than just a few rules.

4. AUTOMATIC RULES COMPOSITION

With automatic rules composition, we start with an initial set of rules for extracting a specific knowledge. Each rule is executed separately over the training set and is saved along with its matching results into a relational database where all training data is stored and relevant results are predefined. Having all this information in a relational database greatly simplifies the analysis of rules results using SQL and data mining tools. It also makes it trivial, using SQL queries, to calculate the TP, TN, FP, and FN of any rule and to compare and contrast the results of different rules.

The biggest benefit of storing rules results is in enabling the automatic generation of the results and the calculation of the F-Score for the combination of any subset from the initial rules without having to run the rules combination over the training set. This allows the evaluation of millions of rules combinations in few seconds instead of spending hours to only run few rules combinations over the training sets. Fig. 7 shows how the results and F-Score of a composite rule (R1 OR R2) can be calculated using the results of its constituent rules. The left part of the array represents the relevant results while the right part represents the irrelevant ones. A 1 in the left part indicates a TP and a 0 in the left part indicates a FN. Similarly, a 1 in the right part indicates a FP and a 0 is for a TN. The example in Fig. 7 also shows how combining two rules can result in a better F-Score.



R1(Precision= 5/8 = 0.63, Recall= 5/10 = 0.5, F-Score= 0.56)

1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0

R2 (Precision= 3/6 = 0.5, Recall= 3/10 = 0.3, F-Score= 0.38)

0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0

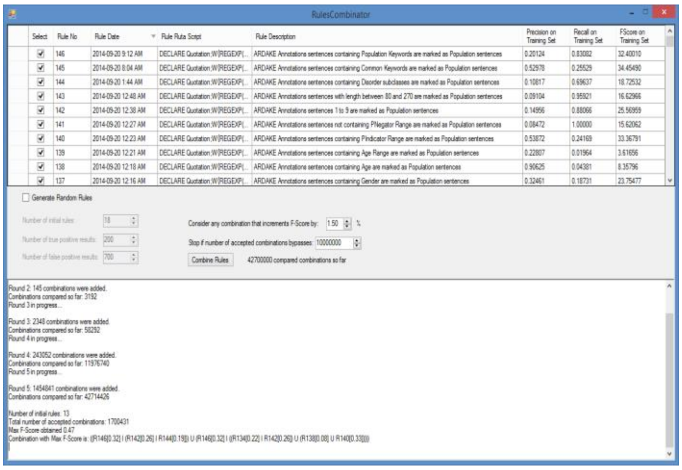
R1 v R2 (Precision= 8/12 = 0.67, Recall= 8/10 = 0.8, F-Score= 0.73)

1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0

Fig.7. Calculating the results of R1 OR R2

Our automatic rules composition tool (Fig. 8) combines each pair of rules in the initial rules set using the AND and the OR logical operators. Composite rules with F-Scores higher than their constituents are added to the initial set of rules and then combined with other primitive and composite rules to get more complex rules with even higher F-Scores. The tool allows users to select the level of granularity for adding composite rules. For example, a user can specify that only rules combinations with an F-Score that is at least 3% higher than their constituent rules are accepted. The process stops when no more rules are added to the rules set or when a specified number of combinations has been generated and tested. The tool returns the rules combination with the highest F-Score.

We simulated thousands of rules combinations scenarios using various number of initial rules at different granularity levels, and different training sets sizes. With the exception of some rare extreme cases, rule combinations always lead to F-Scores that are significantly higher than those of the initial rules (See annex A for sample results).



	Select	Rule No	Rule Date	Rule Ruta Script	Rule Description	Precision on Training Set	Recall on Training Set	FScore on Training Set
	☑	146	2014-09-20 9:12 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing Population Keywords are marked as Population sentences	0.20124	0.83082	32.40010
	☑	145	2014-09-20 8:04 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing Common Keywords are marked as Population sentences	0.52978	0.25529	34.45490
	☑	144	2014-09-20 1:44 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing Disorder subclasses are marked as Population sentences	0.10817	0.69637	18.72532
	☑	143	2014-09-20 12:48 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences with length between 80 and 270 are marked as Population sentences	0.09104	0.95921	16.62966
	☑	142	2014-09-20 12:38 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences 1 to 9 are marked as Population sentences	0.14956	0.88066	25.56959
	☑	141	2014-09-20 12:27 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences not containing PNegator Range are marked as Population sentences	0.08472	1.00000	15.62062
	☑	140	2014-09-20 12:23 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing PIndicator Range are marked as Population sentences	0.53872	0.24169	33.36791
	☑	139	2014-09-20 12:21 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing Age Range are marked as Population sentences	0.22807	0.01964	3.61656
	☑	138	2014-09-20 12:18 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing Age are marked as Population sentences	0.90625	0.04381	8.35796
	☑	137	2014-09-20 12:16 AM	DECLARE Quotation;W{REGEXP(...	ARDAKE Annotations sentences containing Gender are marked as Population sentences	0.32461	0.18731	23.75477

□ Generate Random Rules

Number of initial rules: 18

Number of true positive results: 200

Number of false positive results: 700

Consider any combination that increments F-Score by: 1.50 %

Stop if number of accepted combinations bypasses: 10000000

42700000 compared combinations so far

```
Round 2: 145 combinations were added.
Combinations compared so far: 3192
Round 3 in progress...

Round 3: 2348 combinations were added.
Combinations compared so far: 58292
Round 4 in progress...

Round 4: 243052 combinations were added.
Combinations compared so far: 11976740
Round 5 in progress...

Round 5: 1454841 combinations were added.
Combinations compared so far: 42714426

Number of initial rules: 13
Total number of accepted combinations: 1700431
Max F-Score obtained 0.47
Combination with Max F-Score is: ((R146[0.32] I (R142[0.26] I R144[0.19])) U (R146[0.32] I ((R134[0.22] I R142[0.26]) U (R138[0.08] U R140[0.33]))))
```

Fig. 8. Results-Based rules composer.

5. A RULE-BASED IE PROCESS

We propose using a process inspired from the CRISP-DM methodology for rule-based IE systems. Before creating any rules, it is important to understand the business (domain) for witch the IE system is intended and the data from which the system will extract information and knowledge. Domain experts and data visualization tools can greatly reduce the complexity speed up the first two steps. A training set should be created if it does not already exist. This can be a complex task involving domain experts. When it comes to building and evaluating the model, we suggest the following steps:

1- Look for common and obvious patterns in the positive (i.e. relevant) results of the training set and use a visual rules editor like ARDAKE to build elementary rules that match those patterns.

2- Look for common patterns in the negative (i.e. irrelevant) results of the training set and use ARDAKE or another visual rules editor to update the conditions of the rules defined in step 1 or to create new elementary rules that exclude results containing those patterns.

3- Run each of the previously defined rules individually over the training set and store their outputs for further analysis and rules refinements.

4- Review, compare and contrast the results of your rules. Visualize the results of rules with low F-Score and generate reports to see where improvements can be made. Study the TP, FP, TN, and FN of rules with low F-Score and update or delete some rules if needed. Go back to the previous step if you change any rule.

5- Combine rules based on common elements in their TP and FP sets to create new rules with higher F-Scores. For large rules sets use an automated rules composer like the one we described in section 4.

6. OBSERVATIONS FOR RULES COMPOSITION

The following is a small subset of observations we identified during our rules composition testing. They can be useful for combining rules:

- The intersection of two high-recall, low-precision rules leads to high F-Score (i.e. high-recall, high-precision) when the FP overlap of the two rules is small.
- (R1 AND R2) produces a better precision than R1 if $|TP_{R1} \cap TP_{R2}|/|FP_{R1} \cap FP_{R2}| > |TP_{R1}|/|FP_{R1}|$. This is favorable when the two rules have a big TP overlap and a small FP overlap.
- (R1 AND R2) cannot produce a better recall than R1 because $|TP_{R1} \cap TP_{R2}|$ cannot be $> |TP_{R1}|$
- (R1 OR R2) produces a better precision than R1 if $|TP_{R1} \cup TP_{R2}|/|FP_{R1} \cup FP_{R2}| > |TP_{R1}|/|FP_{R1}|$. This is favorable when the two rules have a small TP overlap and a large FP overlap.
- (R1 OR R2) produces a better recall than R1 if $|TP_{R1} \cup TP_{R2}| > |TP_{R1}|$.
- The granularity level for adding combinations to the rules list has a huge impact on the number of generated rules. The number of combinations to compare increases exponentially with finer granularity levels

7. TESTING OUR RULE-BASED IE PROCESS

We applied the process described in section 5 to create and combine rules that detect sentences containing Population patterns in the NICTA-PIBOSO corpus and obtained interesting results. Thirteen elementary rules were defined and created using ARDAKE after learning the basics of Evidence-Based (EB) medicine and observing the PIBOSO training set for the Population annotations. Some rules were created to identify patterns such as a person's age, an age range and keywords. Other rules are statistical in nature such as the location of annotations within the text and the length range of sentences containing the Population annotations. We also added a semantic rule to annotate any sentence containing a disorder as a Population sentence. This is because we noticed that the majority of Population sentences in the training set of PIBOSO include a disorder to describe the problem of the Patient or the Population. The MeSH ontology was used to define our semantic rule since it has few parent disorder concepts from which all other disorder concepts inherit. Our rules composition tool generated the following formula as the rules combination with the highest F-Score:

$$((R13 \cap (R9 \cap R11)) \cup (R13 \cap ((R1 \cap R9) \cup (R5 \cup R7))))$$

While the F-Scores of individual rules are low, the combination of rules at different granularity levels generated a considerably higher F-Score (36% improvement: 47 for the combination versus 34.54 for the best individual F-Score) as shown in Fig. 8. Note that not all initial 13 rules were needed to produce the highest F-Score as some rules are subsumed by others.

8. CONCLUSIONS AND FUTURE WORK

Information Extraction and Information Retrieval can be rule-based or based on statistical machine learning algorithms. Rule-based systems suffer from the tedious and complex task of creating and maintaining a comprehensive set of rules and determining the rules combination with the highest F-Score. We proposed a highly visual, results-based, process that makes it significantly easier to create and maintain rules for rule-based systems and to automatically generate the rules combination with the highest F-Score. A prototype was developed to prove the effectiveness of our approach. We plan to develop a web in interface for our visual rules editor and rules composer tools to makes them available to more users. We also plan to support multiple simultaneous users working on the same rules set.

REFERENCES

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145-1159.

Chiticariu, L., Li, Y., & Reiss, F. R. (2013). *Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!* Paper presented at the EMNLP. http://dblp.uni-trier.de/db/conf/emnlp/emnlp2013.html#ChiticariuLR13

Cunningham, H., Maynard, D., & Tablan, V. (1999). JAPE: a Java annotation patterns engine.

Fayyad, U. M., Wierse, A., & Grinstein, G. G. (2002). *Information visualization in data mining and knowledge discovery*: Morgan Kaufmann.

Keim, D., Qu, H., & Ma, K.-L. (2014). Big-Data Visualization.

Keim, D. A. (2002). Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, *8*(1), 1-8.

Kluegl, P., Toepfer, M., Beck, P.-D., Fette, G., & Puppe, F. (2014). UIMA Ruta: Rapid development of rule-based information extraction applications. *Natural Language Engineering*, 1-40.

Michaelis, J., & Mönnich, U. (2007). Towards a Logical Description of Trees in Annotation Graphs. *Journal for Language Technology and Computational Linguistics*, *22*(2), 68-83.

Reiss, F., Raghavan, S., Krishnamurthy, R., Zhu, H., & Vaithyanathan, S. (2008). *An algebraic approach to rule-based information extraction.* Paper presented at the Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on.

Toepfer, P. K. M. (2014). UIMA Ruta Workbench: Rule-based Text Annotation. *COLING 2014*, 29.

Appendix A. F-Score for Rules Combinations

The numbers between square brackets show the F-Scores of individual rules.

Initial number of rules: 100
Total number of rules: 948
Max F-Score Id = 307
Max F-Score = 0.80

Rules Combination with maximum F-Score:
(((R13[0.50] U R71[0.54]) I (R90[0.59] U R94[0.54])) U ((R82[0.48] U R84[0.58]) I (R90[0.59] U R94[0.54])))
//

Initial number of rules: 30
Total number of rules: 794
Max F-Score Rule Id = R218
Max F-Score = 0.84

Rules Combination with maximum F-Score:
(((R11[0.31] I R28[0.24]) U (R15[0.28] I R26[0.24])) I (R7[0.50] U ((R9[0.31] I R16[0.37]) U (R20[0.23] I (R4[0.16] I R19[0.18])))))
//

Initial number of rules: 30
Total number of rules: 1512
Max F-Score Rule Id = R596
Max F-Score = 0.91

Rules Combination with maximum F-Score:
((R2[0.54] I R14[0.40]) U ((R14[0.40] I R16[0.43]) U ((R3[0.21] I R5[0.37]) U (R18[0.24] I R21[0.43]))))
//

Initial number of rules: 30
Total number of rules: 18234
Max F-Score Rule Id = R938
Max F-Score = 1.00

Rules Combination with maximum F-Score:
(R10[0.67] U ((R27[0.62] U (R14[0.36] I R26[0.43])) I ((R18[0.36] I R24[0.43]) U (R18[0.36] I R28[0.29]))))
//

Initial number of rules: 20
Total number of rules: 754
Max F-Score Rule Id = R458
Max F-Score = 0.95

Rules Combination with maximum F-Score:
((R8[0.48] I (R12[0.38] U (R0[0.19] I R10[0.22]))) U (R1[0.46] I (R14[0.50] U (R12[0.38] U (R0[0.19] I R10[0.22])))))
//

Initial number of rules: 20
Total number of rules: 64
Max F-Score Rule Id = R56
Max F-Score = 0.82

Rules Combination with maximum F-Score:
(R4[0.32] I (R10[0.55] U (R4[0.32] I (R8[0.24] I R16[0.30]))))