Editorial

IT is my pleasure to present the readers a new issue of the Polibits journal. This issue of Polibits includes 10 papers by authors from 10 different countries: Australia, Chile, Ecuador, Germany, Hungary, Mexico, Portugal, Spain, Sweden, and USA. The majority of the papers included in this issue are devoted to various topics within Artificial Intelligence, probably the widest and most actively growing area of computer science nowadays.

The first five papers of this issue are devoted to one of the most fundamental problem in Artificial Intelligence and in general in science: logical reasoning and knowledge representation.

David Sundgren and Alexander Karlsson from Sweden address a problem of reasoning under uncertainty, which is of great importance in Artificial Intelligence. They analyze the phenomenon of second-order probability: uncertainty, in the form of probability, about the probability of an event. First-order probability reflects our knowledge about an event. For example, suppose we have a fair die; then in the next throwing all numbers are equally probable. Suppose now we have an unfair die that always shows the same number, but we don't know which. Then again, the probability of seeing all numbers at the next throwing are all equal. Now, second-order probabilities reflect our knowledge about the distributions of events. In our example, in the case of a fair die there is only one option: an equal distribution of numbers. However, in the case of unfair die, there are six options: it can always show the number 1, or always the number 2, etc.; since we have no information about this particular die other than that it is unfair, all six options are equally probable. The authors study the uncertainty levels that appear in reasoning with second-order probabilities.

Chaman L. Sabharwal et al. from the USA study intersection of triangles in frame of qualitative spatial reasoning. Triangles are basic shapes used both in mathematics (triangulation in the homology theory) and computer science to represent more complex spatial objects. Detecting intersections between spatial objects is useful for their computational modeling, for example, in CAD / CAM systems. The author describe rather complex logic resulting from intersections of these basic shapes in spatial combinatorics.

Irosh Fernando and Frans A. Henskens from Australia introduce an algorithm for the use of the Select and Test reasoning model in medical expert systems. They give a detailed pseudo-code for their algorithm and even implementation of its most important parts in Java. Their algorithm involves a bottom-up and recursive process with logical inferences, abduction, deduction, and induction. An example small knowledgebase is also given.

Guida Gomes et al. from Portugal analyze various factors and events that can lead to deterioration of buildings and are thus important to know in order to determine a correct strategy for repairing. They use logic programming approach for knowledge representation and reasoning about these events and factors. Specifically, they extend the Eindhoven Classification Model and adapt it to the area of conservation and maintenance of buildings and its causal tree.

Juan Carlos Nieves and Helena Lindgren from Sweden show how to consolidate heterogeneous knowledge sources for decision making and reasoning, for example, about medical diagnosis. They present an algorithm capable to merge deductive and abductive knowledge bases. They explore an argumentation context approach, which follows the way medical professionals typically reason in order to merge two basic kinds of reasoning approaches: deductive and abductive inferences. For this, they introduce two kinds of argumentation frameworks: deductive argumentation frameworks and abductive argumentation frameworks, and merge the corresponding knowledge sources using an approach based on argumentation context systems.

The next two papers show how computer modeling can be usefully applied in economy domain, such as forecasting of economic activity and modeling of economic processes.

Nibaldo Rodriguez and Jose Miguel Rubio L. from Chile and Lida Barba from Ecuador present a forecasting strategy based on stationary wavelet transform combined with radial basis function (RBF) neural network. As a case study, they apply this strategy to improve the accuracy of 3-month-ahead hake catches forecasting of the fisheries industry in the central southern Chile. Their forecasting model decomposes the raw data set into an annual cycle component and an inter-annual component by using 3-levels stationary wavelet decomposition. The components are independently predicted using an autoregressive RBF neural network model. The utility of the proposed model is demonstrated on hake catches data set for monthly periods from 1963 to 2008.

Borja Ponte et al. from Spain address an important economic problem, which is a major concern for companies nowadays: supply chain management. The Bullwhip Effect, related to the amplification of the demand supported by the different levels, is a major cause of inefficiency in the supply chain. The authors present an application of simulation techniques to the study of the Bullwhip Effect in comparison to modern alternatives such as the representation of the supply chain as a network of intelligent agents. They show that the

supply chain simulation is a particularly interesting tool for performing sensitivity analyses in order to measure the impact of changes in a quantitative parameter on the generated Bullwhip Effect. A sensitivity analysis of safety stock illustrates the relationship between Bullwhip Effect and safety stock.

The last three papers are devoted to yet another major area of artificial intelligence: natural language processing. Two of these papers address an emerging and very active area or web opinion mining, and the remaining one introduces a new kind of feature useful for classification tasks.

Melanie Neunerdt et al. from Germany present a technique to train a part-of-speech tagger on real comments left by the users on social media websites. The importance of the analysis of user-contributed contents in social media stems from the huge quantity of such texts, from which user's opinions about products, companies, political parties, or events can be successfully mined. Knowing this information presents better quality of life for consumers, better income for businesses, and real-time democracy for the governments. However, the grammar and style of such texts greatly differs from the grammar and style of traditional sources such as books or newspapers. The majority of existing natural language processing tools is tailored for traditional language and not for web social media. The authors show how a part-of-speech tagger can be trained on real web social media texts. The work described in this paper has received third place best paper award at the 12th Mexican International Conference on Artificial Intelligence, out of 284 submissions from 45 countries.

Grigori Sidorov from Mexico discusses an extension of the syntactic n-gram feature space suggested earlier by him and his co-authors. The extension consists in allowing bifurcations in the traversal of the syntactic tree when forming syntactic n-grams. Syntactic n-grams has been previously shown to be a useful tool in classification tasks such as author identification or plagiarism detection. Numerous examples of forming the syntactic n-grams with bifurcations are given, and a way of their representation in plain text is described.

Finally, István Endrédy and Attila Novák from Hungary present a novel algorithm for removing unimportant information from webpages. Huge body of useful information can be mined from webpages, such as user's opinions about products or political parties. However, analysis of webpages is hindered by a large amount of text and images, such as advertising, formatting, styling, pointers to other articles and webpages, etc., unrelated to the main contents of the webpage and usually automatically added by the webserver. Removing such overhead and leaving only the important contents of webpages for subsequent analysis is a very important practical task. The authors improve over existing algorithms for this task. They also present a new gold standard corpus for evaluation of text cleaning algorithms.

Ildar Batyrshin

Research Professor,
Instituto Mexicano del Petróleo, Mexico
Treasurer,
Mexican Society of Artificial Intelligence

Uncertainty Levels of Second-Order Probability

David Sundgren and Alexander Karlsson

***Abstract*—Since second-order probability distributions assign probabilities to probabilities there is uncertainty on two levels. Although different types of uncertainty have been distinguished before and corresponding measures suggested, the distinction made here between first- and second-order levels of uncertainty has not been considered before. In this paper previously existing measures are considered from the perspective of first- and second-order uncertainty and new measures are introduced. We conclude that the concepts of uncertainty and informativeness needs to be qualified if used in a second-order probability context and suggest that from a certain point of view information can not be minimized, just shifted from one level to another.**

***Index Terms*—Uncertainty, entropy, second-order probability.**

I. Introduction

REASONING under uncertainty is a fundamental problem within artificial intelligence. In this probability is an important tool, but in real life situations there is often uncertainty regarding the probability values themselves. Second-order probability, see e.g. [1], [2], [3], is an hierarchical model of imprecise probability that can be used to model different types of uncertainty regarding first-order probability distributions, e.g., in terms of their quality [4]. Just as in e.g. the possibilistic hierarchy [5], the epistemic reliability model [4] or fuzzy probabilities [6], probability distributions are discriminated by weights. In the case of second-order probability the weights are themselves probabilities. Where there are probability distributions there is uncertainty, with a second-order distribution there is then the uncertainty that comes with the second-order probabilities but also the uncertainty of the first-order probabilities.

Thus it is meaningful to distinguish different types of uncertainty, and in the limits of uncertainty, ignorance and uninformativeness. As is pointed out in [7] ignorance comes in different forms, and E. T. Jaynes wrote in [8] that 'A major thing to be learned in developing this neglected half of probability theory is that the mere unqualified epithet "uninformative" is meaningless.'

A. Levels of Uncertainty in Dempster-Shafer

In the literature on Dempster-Shafer theory [9] there is in e.g. [10] a distinction between two types of uncertainty, *dissonance* and *nonspecificity*. Shannon entropy is in [10] mentioned as an example of a measure of dissonance but not nonspecificity; beliefs expressed in terms of probability distributions are dissonant. Dissonance pertains to probabilistic uncertainty and in e.g. [11] an entropy like measure for dissonance (or discordance) of the basic assignment functions of [9] is introduced. On the other hand nonspecificity is in [10] described as increasing with the number of alternatives in a decision situation and the Hartley measure is put forward as the appropriate measure of nonspecificity. In [12], the measures for discord and nonspecificity are aggregated into *total uncertainty*, see also [13] for a more recent account of uncertainty measures in evidence theory.

Since second-order probability is not equivalent to Dempster-Shafer theory the uncertainty measures designed for belief functions are not directly applicaple to second-order distributions. Yet, in [14] there is a discussion of how second-order distributions could be interpreted in terms of nonspecificty. Smithson [14] recounts the situation in [15] where Miss Julie is invited to bet on the outcomes of three tennis matches in terms of second-order probability. In match A it is known that it will be an even match; in terms of first- and second-order uncertainty there is no second-order uncertainty and maximum first-order uncertainty in match A. Nothing is known about match B and Smithson [14] suggests that the second-order distribution be a uniform distribution spanning the $[0, 1]$ interval, in this case both first- and second-order uncertainty is high but it is questionable whether any second-order distribution can model the ignorance regarding match B. As regards to match C Miss Julie knows that one of the players is excellent and the other an amateur but she does not know which one is the better player, in this case there is no first-order uncertainty but maximum second-order uncertainty. The corresponding second-order distributions could be described as follows in the terms used in the sequel of this paper: In match A we say that the second-order distribution is determined by $p(x_1 = 0.5, x_2 = 0.5) = 1$, in match C we could have the second-order distribution $p(x_1 = 1, x_2 = 0) = 0.5, p(x_1 = 0, x_2 = 1) = 0.5$. The case of match B is probably not possible to fully specify with second-order distributions.

According to Smithson [14] B is the most nonspecific situation, C is more specific than B and A is considered to be "quite specific". With respect to B there are two different approaches; either use a uniform second-order distribution as advised in [14] or refuse to express the uncertainty of B with a second-order distribution. In the first case there is some first-order uncertainty since there is positive probability for high-entropy points. Inasmuch the second-order distribution

Manuscript received on June 27, 2013; accepted for publication on September 30, 2013.

David Sundgren is with the Department of Computer and Systems Sciences, Stockholm University, Sweden (e-mail: dsn@dsv.su.se).

Alexander Karlsson is with the Informatics Research Center, University of Skövde, Sweden (e-mail: alexander.karlsson@his.se).

assigns belief to points that are far apart there is a high second-order uncertainty but belief in neighboring points reduce second-order uncertainty to be less than maximal.

Comparing nonspecificity with first- and second-order uncertainty, if match B after all is represented by a second-order distribution B would be between A and C along both the first- and second-order uncertainty scale while being the most nonspecific, indicating that first-/second-order uncertainty and dissonance/nonspecifity are independent measures. If on the other hand B is left out for being impossible to be modelled with a second-order distribution, dissonance would have positive correlation to first-order uncertainty.

Even though we discuss uncertainty measures for second-order probability rather than for Dempster-Shafer theory there would be a parallell in that what we call first-order uncertainty could be seen as probabilistic just as dissonance is. Second-order level uncertainty could correspondingly described as deterministic.

B. Probabilistic and Deterministic Uncertainty

The distinction might be clarified with an example. Say that we want to express ignorance as to the outcome of an experiment with a second-order probability distribution. One possibility is to assign all second-order probability to the maximum entropy distribution where all outcomes are equally probable. This way we would express ignorance on what could be called the first-order level. But on the second-order level we are absolutely certain, there is no doubt which first-order distribution is the proper one, all uncertainty if placed on the level of first-order probabilities. This situation might be described as being certain of being uncertain. On the other hand we could express ignorance by the uniform second-order distribution on the zero entropy distributions where one of the outcomes is certain. In a first-order perspective there is no uncertainty, distributions with positive entropy are not considered, however we would know no more about the outcome of the experiment. The uncertainty remains but now entirely on the second-order level, you might say that in this case we are uncertain of being certain.

To make the experiment tangible, consider throwing a die. The second-order distribution mentioned first, where all second-order probability is assigned to the uniform distribution $(1/6, 1/6, \ldots, 1/6)$ could be employed to express certainty in that the die is fair. But if a uniform second-order distribution is put on the six zero-entropy distributions $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$ this would mean that we know for certain that the die is fixed to always show the same number but we have no idea which one. A possible interpretation in the realm of philosophy or psychology could be that the first type of second-order distribution could be used by someone who believes in the fundamental randomness of everything but the other type is suitable for an ignorant[1] determinist. These two second-order probability distributions are extreme examples and other second-order distributions could represent mixtures of the two types of ignorance. The point is that ignorance is not enough to specify a second-order distribution unambiguously. And even given the distribution of uncertainty between the two levels it is not obvious how to measure uncertainty.

Mork [16] has performed an extensive study on how uncertainty could be measured from credal sets and from second-order probability distributions. He introduced an entropy based uncertainty measure called GSU (Gärdenfors-Sahlin uncertainty) after [4]. Entropy has also been applied to interval-based imprecise probabilities (i.e., without second-order information) [17]. In this paper we will use entropy as a basis for a majority of our uncertainty measures. By using simple numerical examples, we will contrast the result of our measures to previous measures found in the literature.

In particular, we show that uncertainty measures for second-order probability distributions can be constructed in various ways, and that there are seemingly reasonable requirements on uncertainty measures that are not always met for the measures we discuss. We suggest that uncertainty in a second-order probability setting needs to be qualified in order to be measured consistently. There are likely many aspects with which to specify what is meant by uncertainty for second-order distributions, but whether uncertainty is measured on the first- or second-order level, or both, appears to be a relevant specification. To our knowledge the only previous uncertainty measure for second-order probability that could be said to cover both levels is the GSU measure of [16]. We here introduce a measure for aggregated uncertainty that decomposes naturally into first and second-order levels.

C. Definitions and Notation

Let the outcome space Ω have a finite number of elements n, $\Omega = \{s_i : i = 1, \ldots, n\}$. What we call first-order probabilities are the probabilities of the n outcomes and second-order probability distributions are probability distributions with first-order distributions as random variables. That is, any probability can be seen as a first-order probability, but second-order probability is probabilities over probabilities. Since we are more interested in the probabilities of outcomes rather than the outcomes themselves, we denote the probability $\Pr(s_i)$ by x_i, i.e. all x_i are first-order probability values, $0 \leq x_i \leq 1, \sum_{i=1}^{n} x_i = 1$. All marginals mentioned below are one-dimensional marginal probability distributions.

Further, to simplify computations we restrict the first-order probability values to rational numbers, $x_i = k_i/N$, where $\sum_{i=1}^{n} k_i = N$. A probability distribution will then be considered as a vector $\mathbf{x} = (x_1, \ldots, x_n) = (k_1/N, \ldots, k_n/N)$. Let X denote a set of first-order probabilities $\mathbf{x}$ where the marginals are rational numbers.

[1]The term is of course not used in a derogatory sense.

Furthermore, let us define the set:

$$X_N = \left\{ \mathbf{x} : \mathbf{x} = (k_1/N, \ldots, k_n/N), \sum_{i=1}^{n} k_i = N, k_i \in \mathbb{N} \right\}$$

i.e., the set of all first-order probabilities in the form of rational numbers that fulfill $\sum_{i=1}^{n} k_i = N$. For every N, X_N is finite so we are therefore restricted to discrete second-order probability distributions over first-order distributions. That is, the discrete second-order probability distributions discussed here have X_N as outcome space. Second-order probability distributions are denoted p, and the probability of first-order probability distribution $\mathbf{x} \in X_N$ is then $p(\mathbf{x})$. The marginal probability of first-order probability value x_i is written $p_i(x_i)$. We remind of the definition of Shannon entropy

$$H(\mathbf{x}) = -\sum_{i=1}^{n} x_i \log_2 x_i \,.$$

II. First-Order Uncertainty

As we have argued we may distinguish two levels of uncertainty. The term first-order uncertainty is intended to capture uncertainty on the level of first-order probabilities in the context of a second-order probability distribution. In other words, first-order uncertainty is the type of uncertainty expressed by probabilities of unknown outcomes of an event. In the absence of second-order probability an uncertainty measure such as entropy would be used, but here we have second-order probability distributions to account for. In this section we present one way of measuring such first-order uncertainty.

Weighted entropy. This uncertainty measure intends to capture the collected amount of entropy in the first-order probability values. Since a second-order distribution assigns probability values to the first-order variables, the entropy values are weighted accordingly. That is, the more probable a vector $\mathbf{x}$ is, the more weight we give to its entropy.

$$W_H(X,p) = \sum_{\mathbf{x}\in X} p(\mathbf{x}) H(\mathbf{x}) = -\sum_{\mathbf{x}\in X} p(\mathbf{x}) \sum_{i=1}^{n} x_i \log_2 x_i \,.$$

III. Second-order Uncertainty

These measures are meant give the degree of uncertainty on the second-order level. For instance, if all second-order probability is given to low entropy points there could still be a high degree of uncertainty in that there is little or no commitment to any one particular outcome.

We have found three basic approaches to second-order uncertainty, one is the entropy of the second-order probability distribution, the other one is based on how much the second-order distribution is spread out. The latter, distance-based measures are justified by the intuition that uncertainty could be expressed by conflicting statements. The third measure considers the volume of the support of the second-order distribution.

Entropy. This measure is simply the entropy $H(p)$ of the second-order distribution:

$$H(X,p) = -\sum_{\mathbf{x}\in X} p(\mathbf{x}) \log_2 p(\mathbf{x}) \,.$$

Weighted Kullback-Leibler divergence. If we want to be able to compare second- and first-order uncertainty, we have to use an entropy-based distance measure so that first- and second-order uncertainty are measured in the same units. If second-order uncertainty is linked to the spread of belief over the probability simplex and measure entropy, we suggest the average Kullback-Leibler divergence, see [18], to the mean as measure of second-order uncertainty, i.e., the mean of the second-order distribution $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)$, defined by $\mu_i = \sum_{\mathbf{x}\in X_N} p(\mathbf{x}) x_i$ is used as point of reference.

$$W_{D_{\mathrm{KL}}}(X,p) = \sum_{\mathbf{x}\in X} p(\mathbf{x}) D_{\mathrm{KL}}(\mathbf{x}|\boldsymbol{\mu}) = \sum_{\mathbf{x}\in X} p(\mathbf{x}) \sum_{i=1}^{n} x_i \log_2 \frac{x_i}{\mu_i} \,.$$

Degree of imprecision. The degree of imprecision [19] aims to be an approximation of the hypervolume spanned by the first-order probability distributions with positive second-order probability.

$$DI(X,p) = \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\mathbf{x}\in \hat{X}_N} x_i - \min_{\mathbf{x}\in \hat{X}_N} x_i \right],$$

where $\hat{X}_N = \{\mathbf{x} \in X | p(\mathbf{x}) > 0\}$. Note that the computation of DI can be performed by maximizing respective minimizing over the extreme points of the convex hull of first-order probability distributions with positive second-order probability.

IV. Aggregated Uncertainty

Under this rubric we collect measures that could claim to express the aggregated degree of uncertainty on first and second-order level.

Mork's GSU. The uncertainty measure introduced in [16] is named after Gärdenfors and Sahlin and is inspired by their discussion in [4] about epistemic reliability. The idea might be summarized in that the information value is reduced by adding second-order probability through a convex combination of information values.

The measure is defined by:

$$GSU(X,p) = \max_{\mathbf{x}\in X} \left\{ -\sum_{i=1}^{n} \left[x_i \sum_{\mathbf{x}\in X} p(\mathbf{x}) \log_2 x_i \right] \right\} .$$

Sum of W_H and $W_{D_{\mathrm{KL}}}$. Since both the first-order measure of weighted entropy and the second-order measure of weighted Kullback-Leibler divergence are based on entropy of first-order probabilities and have the same units the sum of the measures is meaningful.

TABLE I
NUMERICAL EXAMPLES

Examples	First- and Second-order Probability Distributions
A	$X = \{(1/3, 1/3, 1/3)\}$, $p(1/3, 1/3, 1/3) = 1$
B	$X = \{(32, 29, 29)/90, (29, 32, 29)/90, (29, 29, 32)/90\}$, $\forall \mathbf{x} \in X, p(\mathbf{x}) = 1/3$
C	$X = X_{10}, (\forall \mathbf{x} \in X)(Pol(\mathbf{x}\|(1/3, 1/3, 1/3)))$ (Perks' prior)
D	$X = X_{10}, (\forall \mathbf{x} \in X)(Pol(\mathbf{x}\|(1/2, 1/2, 1/2)))$ (Jeffreys' prior)
E	$X = X_{10}, (\forall \mathbf{x} \in X)(Pol(\mathbf{x}\|(1, 1, 1)))$ (Bayes-Laplace's prior)
F	$X = \{(1/6, 1/6, 2/3), (1/6, 2/3, 1/6)\}$, $p(1/6, 1/6, 2/3) = 3/4, p(1/6, 2/3, 1/6) = 1/4$
G	$Y = \{(2/3, 1/6, 1/6), (1/3, 1/3, 1/3), (1/4, 1/4, 1/2)\}$, $\forall \mathbf{y} \in Y, p(\mathbf{y}) = 1/3$
H	$(X, Y) = X \times Y = \{(\mathbf{x}, \mathbf{y})\|\mathbf{x} \in X, \mathbf{y} \in Y\}$, $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$, $p(\mathbf{x}), p(\mathbf{y})$ from F and G.
I	$(X, Y) = X \times Y = \{(\mathbf{x}, \mathbf{y})\|\mathbf{x} \in X, \mathbf{y} \in Y\}$, $\forall (\mathbf{x}, \mathbf{y}) \in X \times Y p(\mathbf{x}, \mathbf{y})) = 1/6$, $p(\mathbf{x}), p(\mathbf{y})$ from F and G.
J	$X = \{(1/6, 1/6, 2/3), (1/6, 2/3, 1/6), (7/10, 1/10, 1/5)\}$, $p(\mathbf{x_1}) = 1/2, p(\mathbf{x_2}) = 1/3, p(\mathbf{x_3}) = 1/6$.
K	X as in J, but $\forall \mathbf{x} \in X, p(\mathbf{x}) = 1/3$.

Interestingly, the sum of W_H and $W_{D_{\mathrm{KL}}}$ equals the entropy of the second-order distribution's mean:

Theorem 1: $W_H(X, p) + W_{D_{\mathrm{KL}}}(X, p) = H(\boldsymbol{\mu})$, where $\mu_i = \sum_{\mathbf{x} \in X} p(\mathbf{x}) x_i$.

Proof:

$$W_H(X, p) + W_{D_{\mathrm{KL}}}(X, p) =$$
$$\sum_{\mathbf{x} \in X} p(\mathbf{x}) \sum_{i=1}^{n} x_i \left(\log_2 \frac{x_i}{\mu_i} - \log_2 x_i \right) =$$
$$- \sum_{\mathbf{x} \in X} p(\mathbf{x}) \sum_{i=1}^{n} x_i \log_2 \mu_i =$$
$$- \sum_{i=1}^{n} \left(\sum_{\mathbf{x} \in X} p(\mathbf{x}) x_i \right) \log_2 \mu_i = - \sum_{i=1}^{n} \mu_i \log_2 \mu_i = H(\boldsymbol{\mu})$$

■

V. NUMERICAL EXAMPLES

To better understand these uncertainty measures we have applied them to some second-order probability distributions with various intuitive uncertainty properties. Some of the distributions have only a few points, other distributions have support on the entire space X_N, for $N = 10$. The latter distributions will come from the multivariate Pólya family [20]:

$$Pol(\mathbf{k}|\boldsymbol{\alpha}) = \frac{N! \Gamma\left(\sum_{i=1}^{n} \alpha_i\right)}{\Gamma\left(N + \sum_{i=1}^{n} \alpha_i\right)} \prod_{i=1}^{n} \frac{\Gamma(k_i + \alpha_i)}{k_i! \Gamma(\alpha_i)} ,$$

where $\sum_{i=1}^{n} k_i = N$ and α_i are parameters of the corresponding Dirichlet distribution. Note that we can obtain a distribution over X_N by $Pol(\mathbf{k}/N|\boldsymbol{\alpha})$. The Pólya family of distributions can be seen as the discrete counterpart of the Dirichlet family and is the result of integrating out the underlying probabilities drawn from a Dirichlet distribution in a multinomial distribution.

Consider the examples defined by Table I (where $n = 3$) and the results of applying the uncertainty measures, shown in Table II.

The purpose of examples A and B is to how different measures deal with two distributions that put all second-order probability on high entropy points; on the one hand A where the second-order distribution has support only on the maximum entropy point, on the other B where there is a uniform distribution on three different points that are close to the maximum entropy. In Table II we see that H, the entropy of the second-order distribution is the only measure that makes much of the fact that the second-order distribution has support on three points rather than one.

In examples C, D and E we look at symmetric Pólya distributions with parameters $1/3$ (Perks), $1/2$ (Jeffreys) and 1 (Bayes-Laplace), respectively. The first-order level measure W_H and the second-order level measure H increase with the Dirichlet parameters while the other second-order level measure $W_{D_{\mathrm{KL}}}$ decreases. If we interpret the Dirichlet parameters as a measure of the amount of available data higher parameter values would give more (first order-) probabilistic credibility to the mean probability vector.

If the parameters are equal, the mean would be the maximum entropy point of the simplex. In our examples then, first-order uncertainty would increase (as does W_H), and second-order uncertainty decrease (since there is more data to support a particular first-order probability).

We may also note that GSU gives infinite values in these cases. This is because there are zero-valued first-order probabilities with positive second-order probability. It could be argued that zero-valued first-order probabilities x_i should be excluded, since the event that x_i is the probability of is impossible and does not belong in the outcome space. On the other hand it is feasible that (on a second-order level) it is possible but not certain that an event can not occur, i.e. that neither $x_i = 0$ or $x_i > 0$ can be ruled out.

TABLE II
RESULTS OF APPLYING THE UNCERTAINTY MEASURES TO THE EXAMPLES

	First-order	Second-order			Aggregated	
Examples	W_H	DI	H	$W_{D_{\mathrm{KL}}}$	GSU	$H(\mu)$
A	1.5850	0.0000	0.0000	0.0000	1.5850	1.5850
B	1.5834	0.0333	1.5850	0.0016	1.5865	1.5850
C	0.6852	0.5682	5.5146	0.8997	∞	1.5850
D	0.8285	0.5682	5.8399	0.7565	∞	1.5850
E	1.0486	0.5682	6.0444	0.5364	∞	1.5850
F	1.2516	0.5000	0.8113	0.1768	2.0016	1.4284
G	1.4455	0.2500	1.5850	0.1091	1.7233	1.5546
H	2.6972	0.5000	2.3962	0.2858	3.7249	2.9830
I	2.6972	0.5000	2.5850	0.3408	3.4749	3.0379
J	1.2358	0.5333	1.4591	0.3189	2.0803	1.5547
K	1.2200	0.5333	1.5850	0.3633	2.0635	1.5833

VI. PROPERTIES

In [16] there are sets of requirements for uncertainty measures both for credal sets and for second-order probability distributions. Likewise there is in [21] a list of requirements for measures of uncertainty, but designed for belief functions (see [9]). Since the requirements of [21] do make sense when translated to a second-order probability setting we will consider also these, some of them also coincide with the requirements of [16]. Please note that since belief functions in the language of second-order probability is best translated as lower bounds of first-order probabilities that in turn do not determine unique second-order distributions, the translation of properties must at times be ad hoc. Strictly speaking then, the requirements below that are taken from [21] is to be considered as inspired by [21] rather than literal translations.

Below we describe these requirements briefly, as far as they can be expressed in the terms used in this paper. The original authors have different notation, and in the case of [21] information is carried by belief functions instead of second-order distributions. Let $\mathcal{U}(X,p)$ denote an uncertainty measure for a second-order probability distribution p with support on X.

(i) *Conicides with entropy* C1 in [16], (1) in [21]. Uncertainty coincides with entropy if all second-order probability is put on a single vector. That is, If $p(\mathbf{x}) = 1$ for some $\mathbf{x} \in X$, then $\mathcal{U}(X,p) = H(\mathbf{x})$.

(ii) *Continuous* C2 in [16]. $\mathcal{U}$ is continuous in p.

(iii) *Symmetric* C3 in [16]. $\mathcal{U}$ is symmetric, i.e. invariant under permutations in the vectors $\mathbf{x}$, i.e.. if $Y = \{(x_{\pi(1)}, x_{\pi(2)}, \ldots x_{\pi(n)}) | \mathbf{x} \in X\}$ where π is a permutation of $\{1,2,\ldots,n\}$, $\mathcal{U}(X,p) = \mathcal{U}(Y,p)$.

(iv) *Hartley* (2) in [21]. For a uniform second-order distribution, i.e. p s.t. $p(\mathbf{x}) = 1/|X|$ for $\mathbf{x} \in X$, uncertainty equals $\mathcal{U}(X,p) = \log_2 |X|$, the Hartley measure of X.

(v) *Range* (3) in [21]. The range of $\mathcal{U}$ is the interval $[0, \log_2 |\Omega|] = [0, \log_2 n]$.

(v′) With an alternative interpretation, (v)' requires the range to be $[0, \log_2 |X|]$.

(vi) *Additive* C4 in [16], (5) in [21]. Additivity, i.e. if X is the cartesian product $A \times B$ and A and B are independent so that $p((\mathbf{x},\mathbf{y})) = p(\mathbf{x})p(\mathbf{y})$, then $\mathcal{U}(X,p) = \mathcal{U}(A,p) + \mathcal{U}(B,p)$.

(vii) *Subadditive* C5 in [16], (4) in [21]. Subadditivity, i.e. if X is the cartesian product $A \times B$, then

(viii) *Bounded by entropy* NC1' in [16]. The uncertainty of a second-order distribution is at least as high as the entropy of any of the first-order probability distributions in its support. $\mathcal{U}(X,p) \geq \max_{\mathbf{x} \in X} H(\mathbf{x})$.

(ix) *Bounded by credal set* NC3 in [16]. If Π is a partition of X and $\mathrm{Conv}(X)$ is the convex hull of X, then $\mathcal{U}'(\Pi(\mathrm{Conv}(X))) \geq \mathcal{U}(\Pi(X),p)$, where $\mathcal{U}'$ is the corresponding uncertainty measure for a credal set.

(ix′) For some of the measures considered here ($W_H, H, W_{D_{\mathrm{KL}}}, H(\boldsymbol{\mu})$) it is not possible to remove the second-order distribution p. But we might formulate a version of NC3 that retains some of what we believe is intended by the requirement. Short of removing p we replace p with the maximum entropy second-order distribution. We declare requirement (ix)' to be that $\mathcal{U}(X,p) \leq \mathcal{U}(X,q)$, where q is the uniform second-order distribution on X.

In Table III we summarize our findings of whether the measures studied here fulfill the requirements. $H(\boldsymbol{\mu})$ in the rightmost column refers to the sum of W_H and $W_{D_{\mathrm{KL}}}$.

VII. SUMMARY AND CONCLUSIONS

In this paper, we have suggested a division of uncertainty between two levels corresponding to first and second-order probability. With such a division it becomes possible to distinguish between on the one hand uninformativeness in the guise of a uniform probability distribution over the possible outcomes and on the other hand uninformativeness in the form of a uniform second-order distribution. In the first case we are sure of being uncertain but in the other we express uncertainty on a higher level.

We studied the behavior of six different uncertainty measures, two for each uncertainty level and two for

TABLE III

PROPERTIES OF THE UNCERTAINTY MEASURES. NOTES: [1] $W_H + W_{D_{\mathrm{KL}}}$ FULFILLS REQUIREMENT (IV) IF AND ONLY IF $\mu_i = 1/n$ FOR $i = 1, \ldots, n$. [2] EXAMPLE A IN TABLE II SERVES AS COUNTER EXAMPLE IN THE NEGATIVE CASES. [3] PROVED IN [16]. [4] EXAMPLES F, G AND H IN TABLE II SERVE TOGETHER AS COUNTER EXAMPLE SINCE THE UNCERTAINTIES OF ROW F AND G SHOULD ADD TO THE VALUE IN H. [5] WE HAVE NOT FOUND NEITHER PROOF NOR COUNTEREXAMPLE. [6] EXAMPLES F, G AND I IN TABLE II SERVE TOGETHER AS COUNTER EXAMPLE SINCE THE SUM OF THE UNCERTAINTIES OF ROW F AND G SHOULD NOT BE LESS THAN THE VALUE IN I. [7] NOT APPLICABLE SINCE WE HAVE FOUND NO WAY OF DEFINING A CORRESPONDING MEASURE FOR A CREDAL SET, I.E. WITH OUT THE SECOND-ORDER PROBABILITY DISTRIBUTION. [8] SEE EXAMPLES J AND K.

		First-order	Second-order			Aggregated	
	Requirement	W_H	DI	H	$W_{D_{\mathrm{KL}}}$	GSU	$H(\boldsymbol{\mu})$
(i)	*Coincides with entropy*	Yes	No	No	No	Yes	Yes
(ii)	*Continuous*	Yes	Yes	Yes	Yes	Yes[3]	Yes
(iii)	*Symmetric*	Yes	Yes	Yes	Yes	Yes[3]	Yes
(iv)	*Hartley*	No	No	Yes	No	No	No[1]
(v)	*Range*	Yes	Yes	No	Yes	No	Yes
(v′)		No[2]	No[2]	Yes	No[2]	Yes	No[2]
(vi)	*Additive*	Yes	No[4]	Yes	Yes	Yes[3]	Yes
(vii)	*Subadditive*	?[5]	?[5]	No[6]	No[6]	?[5]	No[6]
(viii)	*Bounded by entropy*	No	No	No	No	Yes[3]	No
(ix)	*Bounded by credal set*	N. a.[7]	Yes	N. a.[7]	N. a.[7]	Yes[3]	N. a.[7]
(ix′)		No[8]	Yes	Yes	?[5]	No[8]	No

the aggregated uncertainty. We introduced a new measure for aggregated uncertainty, the sum of weighted entropy and Kullback-Leibler divergence, where the weights are second-order probabilities.

Furthermore, we showed that such a measure is equivalent to the entropy of the mean first-order probability distribution. In the paradigm of two levels of uncertainty, the entropy of the mean could be viewed as the amount of uncertainty that can be distributed on the first and second-order levels. From such a perspective it is impossible to unequivocally express unqualified ignorance, i.e. one must declare how the uncertainty is distributed. And unless some reasonable principle of uninformative distribution of uncertainty is formulated we cannot say that first-order uncertainty is more or less uncertain than second-order uncertainty.

We compared the measures by a set of properties that has previously been utilized by [16] and [21]. Our separation of measures into first, second-order and aggregated is not very well reflected in the properties that are held, there is no apparent pattern distinguishing the groups of measures. The relation of measures and properties could be seen as a degree of quality for the measures, the more properties held the better. On the other hand it could be discussed how appropriate the properties are for first and second-order measures of uncertainty.

For example property (viii) seems at odds with the idea that a second-order probability distribution weighs first-order distributions differently. From our perspective it is perfectly reasonable that an uncertainty measure for a second-order distribution is lower than the entropy of any first-order distribution in its support, at least if this distribution has low second-order probability. If a distributions' second-order probability is low then it and its properties have a correspondingly low impact.

If we compare our aggregated measure with the GSU-measure, several differences can be found. Perhaps the first notably such is that GSU gives ∞ as a result for examples C - E in Table II while our measure are always finite. As previously mentioned, the reason for such a result is due to that a positive second-order probability was assigned to a zero first-order probability. Beside the differences in properties, found in Table III, the GSU-measure also seems to put more emphasis on the first-order level, as is seen from examples A and B.

In example A there is no uncertainty at the second level but a maximum of uncertainty on the first level, while in example B the second-order uncertainty is uniformly distributed around this maximal first-level uncertainty. In this case DI and the GSU-measure gives a aggregated uncertainty that is a bit higher in example B whilst our measure gives the same amount of uncertainty, however, distributed differently among the first and second-order level. As for examples A and B, the pure entropy measure H stands out since it measures the uncertainty of Example A as zero but the uncertainty of B as the maximal $\log_2 3$. This example shows that it could be problematic to use a pure second-order probability entropy measure unless you do believe that belief in three different but close points should reflect much higher uncertainty than belief in a single point. H is also unique among the measures studied here in that it is bounded by the number of first-order distributions in the support of the second-order distributions, giving much higher uncertainty values for Examples C, D and E than the others except GSU.

We have introduced a new dimension of uncertainty but many questions remain. What should be required of measures for first- and second-order uncertainty of second-order distributions? Is it possible to express a larger amount of uncertainty with higher-order distributions? And how is uncertainty on higher levels to be measured?

The assumption of discrete distributions causes many questions. See for instance the discussion above about the measure H, does a finer granularity imply greater uncertainty?

The points in the real-valued probability simplex that have no support in our discrete examples, should they be viewed as non-existent or simply as (second-order) impossible? Under what circumstances can discrete second-order distributions be seen as corresponding to reality, or when can they be justified as approximations or simplifications? And what consequences do such considerations have on the properties that are desired for uncertainty measures? And how could uncertainty measures for continuous second-order distributions be designed and interpreted? These and similar questions are to be addressed in future research.

REFERENCES

[1] R. F. Nau, "Uncertainty aversion with second-order utilities and probabilities," *Management Science*, vol. 52, no. 1, pp. 136–145, 2006.

[2] L. V. Utkin and T. Augustin, "Decision making with imprecise second-order probabilities," in *ISIPTA '03 - Proceedings of the Third International Symposium on Imprecise Probabilities and Their Applications*, 2003, pp. 547–561.

[3] L. Ekenberg and J. Thorbiörnson, "Second-order decision analysis," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 9, No 1*, vol. 9, no. 1, pp. 13–38, 2001.

[4] P. Gärdenfors and N.-E. Sahlin, "Decision, probability and utility: Selected readings." in *Decision, Probability and Utility: Selected Readings*. Cambridge University Press, 1988, ch. 16, Unreliable probabilities, risk taking, and decision making, pp. 313–334.

[5] G. D. Cooman and P. Walley, "A possibilistic hierarchical model for behaviour under uncertainty," *Theory and Decision 52 (4)*, pp. 327–374, 2002.

[6] L. A. Zadeh, "Fuzzy probabilities," *Information Processing and Management, 20*, pp. 363–372, 1984.

[7] P. Smets, "Varieties of ignorance and the need for well-founded theories," *Inf. Sci.*, pp. 135–144, 1991.

[8] E. T. Jaynes, "Monkeys, kangaroos, and n," *Maximum Entropy and Bayesian methods in Applied Statistics: proceedings of the 4th Maximum Entropy Workshop*, pp. 26–58, 1984.

[9] G. Shafer, *A Mathematical theory of evidence*. Princeton University Press, 1976.

[10] G. J. Klir, "A principle of uncertainty and information invariance," *International Journal Of General System*, vol. 17, no. 2-3, pp. 249–275, 1990.

[11] R. R. Yager, "Entropy and specificity in a mathematical theory of evidence," *International Journal of General System*, vol. 9, no. 4, pp. 249–260, 1983.

[12] G. J. Klir and R. M. Smith, "Recent developments in generalized information theory," *International journal of fuzzy systems*, vol. 1, no. 1, pp. 1–13, 1999.

[13] A.-L. Jousselme, C. Liu, D. Grenier, and E. Bosse, "Measuring ambiguity in the evidence theory," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 36, no. 5, pp. 890–903, 2006.

[14] M. Smithson, "Freedom: A measure of second-order uncertainty for intervalic probability schemes," in *Proceedings of the Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-89)*. Corvallis, Oregon: AUAI Press, 1989, pp. 327–334.

[15] P. Gärdenfors and N.-E. Sahlin, "Unreliable probabilities, risk taking, and decision making," *Synthese*, vol. 53, no. 3, pp. 361–386, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF00486156

[16] J. C. Mork, "Uncertainty, credal sets and second order probability," *Synthese*, 2011, qP 20120202.

[17] A. Bronevich and A. Lepskiy, "Measuring uncertainty with imprecision indices," in *ISIPTA '07 - Proceedings of the Fifth International Symposium on Imprecise Probabilities and Their Applications*, 2007, pp. 47–56.

[18] S. Kullback and R. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[19] A. Karlsson, R. Johansson, and S. F. Andler, "Characterization and empirical evaluation of bayesian and credal combination operators," *Journal of Advances in Information Fusion*, vol. 6, pp. 150–166, 2011.

[20] G. Pólya, "Sur quelques points de la théorie des probabilités," *Ann. Inst. Poincaré*, vol. 1, pp. 117–161, 1931.

[21] D. Harmanec and G. J. Klir, "Measuring total uncertainty in dempster-shafer theory: A novel approach," *International Journal of General Systems*, vol. 22, no. 4, pp. 405–419, 1994.

Triangle-Triangle Intersection Determination and Classification to Support Qualitative Spatial Reasoning

Chaman L. Sabharwal, Jennifer L. Leopold, Douglas McGeehan

Abstract—***In CAD/CAM modeling, objects are represented using the Boundary Representation (ANSI Brep) model. Detection of possible intersection between objects can be based on the objects' boundaries (i.e., triangulated surfaces), and computed using triangle-triangle intersection. Usually only a cross intersection algorithm is needed; however, it is beneficial to have a single robust and fast intersection detection algorithm for both cross and coplanar intersections. For qualitative spatial reasoning, a general-purpose algorithm is desirable for accurately differentiating the relations in a region connection calculus, a task that requires consideration of intersection between objects. Herein we present a complete uniform integrated algorithm for both cross and coplanar intersection. Additionally, we present parametric methods for classifying and computing intersection points. This work is applicable to most region connection calculi, particularly VRCC-3D+, which detects intersections between 3D objects as well as their projections in 2D that are essential for occlusion detection.***

***Index Terms*—Intersection detection, classification predicates, spatial reasoning, triangle-triangle intersection.**

I. Introduction

THERE are relatively few software applications supporting qualitative spatial reasoning. In part, this may be due to the complexity in determining the intersection between 2D/3D objects. Yet the ability to detect the existence of a possible intersection between pairs of objects can be important in a variety of problem domains such as geographic information systems [1], CAD/CAM geometric modeling [2], real-time rendering [3], geology [4], networking and wireless computing.

In qualitative reasoning, it is not necessary to know the precise intersection between pairs of objects; it is sufficient to detect and classify the intersection between objects. Typically, the boundary of each object is represented as a triangulated surface and a triangle-triangle intersection is the computational basis for determining intersection between objects. Since an object boundary may contain thousands of triangles, algorithms to speed up the intersection detection process are still being explored for various applications, sometimes with a focus on innovations in processor architecture [5, 6, 7].

For pairs of triangles, there are three types of intersections: zero dimensional (single point), one-dimensional (line segment), and two dimensional (area) intersection. In the past, almost all attention has been devoted to determining the cross intersections, which resulted in an absence of analysis in two-dimensional intersections. Coplanar triangle intersections are unique because an intersection may be any of the aforementioned three types. If the triangles cross-intersect, only zero or one-dimensional intersection is possible. If the planes are parallel and distinct, the triangles do not intersect. If the triangles are coplanar, then there is a possibility of intersection. Even when the cost of intersecting a triangle pair is constant, the cost of intersecting a pair of objects A and B is order $O(T_A \times T_B)$ where T_A is the number of triangles in object A, and T_B is the number of triangles in object B.

In qualitative spatial reasoning, spatial relations between regions are defined axiomatically using first order logic [8] or the 9-Intersection model [9]. Using the latter model, the spatial relations are defined using the intersections of the interior, boundary, and exterior of one region with those of a second region. It has been shown in [10] that it is sufficient to define the spatial relations by computing 4-Intersection predicates, (namely, Interior–Interior (IntInt), Boundary–Boundary (BndBnd), Interior–Boundary (IntBnd), and Boundary–Interior (BndInt)) instead of 9-Intersections.

Since IntBnd and BndInt are the converse of each other, only three algorithms are necessary for these predicates. In order to implement these algorithms, we must first solve the triangle-triangle intersection determination, as it is a lower level problem that must be solved in order to determine the 4-Intersection predicates that, in turn, determine the qualitative spatial relation between two objects.

This paper is organized as follows: Section II briefly reviews the background and related cross intersection framework. Section III discusses motivation and conceptual classification of intersections, whereupon Section IV develops the overall main algorithm for triangle-triangle intersection. Section V describes the area intersection algorithm for general

Manuscript received May 25, 2013. Manuscript accepted for publication September 30, 2013.

The authors are with the Missouri University of Science and Technology, Rolla, Missouri, 65409, USA (email: {chaman, leopoldj, djmvfb}@mst.edu).

triangles, and predicates for classifying the intersection between pairs of triangles, after which Section VI discusses the applications to qualitative spatial reasoning.

II. Background

A. The Traditional Algorithm

Many papers have been written on the intersection between a pair of triangles [3, 11, 12, 13, 14, 15]. Interestingly, most of them simply reinvent the algorithm and implement it slightly differently and more efficiently, with no innovation. A recent paper [7] surveyed various approaches for determining the cross intersection detection, and developed a fast vector version of the cross intersection detection, as well as classification of the type of intersection. Our approach is exhaustive, integrating both cross and coplanar intersection, and analytically more rigorous than the previous approaches [3, 11]. It is described in the next section where we follow the approach similar to the techniques used in [7] for cross intersection. The cross-intersection standalone algorithm is described as follows:

***boolean triTriCrossInt* (*tr*1 = *ABC, tr*2 = *PQR*)**

input: two triangles whose planes cross intersect

output: true if the triangles intersect, else false

The vector equations for two triangles ABC and PQR are

$$R_1(u, v) = A + u\,U + v\,V,\ 0 \le u, v, u + v \le 1$$
$$R_2(s, t) = P + s\,S + t\,T,\ 0 \le s, t, s + t \le 1,$$

where $U = B - A$, $V = C - A$, and $S = Q - P$, $T = R - P$.

Let $N_1 = U \times V$, $N_2 = S \times T$ be normals to the planes supporting the triangles directed away from the objects. The triangles intersect if there exist some barycentric coordinates (u, v) and (s, t) satisfying the equation

$$A + u\,U + v\,V = P + s\,S + t\,T$$

Since $N_1 \times N_2 \neq 0$ for cross intersecting triangles, and S and T are orthogonal to N_2, the dot product of this equation with N_2 eliminates S and T from the above equation to yield

$$u\,U \bullet N_2 + v\,V \bullet N_2 = AP \bullet N_2$$

This is the familiar equation of a line in the *uv*-plane for real variables u, v. The vector equation using real parameter λ becomes

$$(u, v) = AP \bullet N_2 \frac{(U \bullet N_2, V \bullet N_2)}{U \bullet N_2^{\,2} + V \bullet N_2^{\,2}} + \lambda (V \bullet N_2, -U \bullet N_2)$$

Then parameter values u, v are explicitly written as

$$u = AP \bullet N_2 \frac{U \bullet N_2}{U \bullet N_2^{\,2} + V \bullet N_2^{\,2}} + \lambda V \bullet N_2$$

$$v = AP \bullet N_2 \frac{V \bullet N_2}{U \bullet N_2^{\,2} + V \bullet N_2^{\,2}} - \lambda U \bullet N_2$$

$$u + v = AP \bullet N_2 \frac{(U \bullet N_2 + V \bullet N_2)}{U \bullet N_2^{\,2} + V \bullet N_2^{\,2}} + \lambda (V \bullet N_2 - U \bullet N_2)$$

If there is a λ in these three equations such that $0 \le u, v, u + v \le 1$, the triangles are ensured to intersect. The range of values of λ is bounded by λ_m and λ_M. This detects whether the two triangles cross intersect only.

In fact, for precise intersection, using λ_m, λ_M, as parameter values, we compute (u_m, v_m) and (u_M, v_M) for the segment of intersection on ABC. Similarly the values (s_m, t_m) and (s_M, t_M) represent the segment of intersection on PQR. The precise intersection between the two triangles is the common segment of these two segments. If the segment degenerates into a single point, the parameter values also can be used to classify the intersection as a vertex, an edgeInterior point or triangleInterior point in the triangle ABC.

III. Classification of Triangle Intersections

For spatial reasoning, we detect intersection between pairs of 2D/3D objects and classify pairwise intersection predicates IntInt, IntBnd, BndInt, and BndBnd, without computing the extent of intersections. The cross intersection can be characterized into seven categories [7]. When cross intersection is insufficient to determine tangential intersection, some applications such as RCC8 and VRCC-3D$^+$ [6] resort to coplanar intersection to support relations such as externally connected (EC) and tangentially connected (TPP, TPPc).

The precise intersection of coplanar triangles is a little more complex because it can result in area intersection as well; the coplanar triangles intersection can be classified as: Single Point Intersection (*vertex-vertex, vertex-edgeInterior*), Line Segment Intersection (*edge-edgeCollinear*), Area Intersection bounded by 3, 4, 5, 6 edges, (Fig. 4, Fig. 5(a, b, c)). A triangle may be entirely contained in the other triangle (Fig. 5(d)). In this paper, we present a detailed analytical study of the intersection of coplanar triangles, which has not been previously presented.

The intersection between a pair of triangles can be abstracted as Cross (C) intersection or Parallel (P) coplanar triangles intersection. For taxonomy of cross and parallel coplanar triangles, the conceptual intersections are supported with figures presented here. The specific cases are as follows:

No intersection

disjoint (C, P) (see Fig. 1)

Single Point Intersection

vertex-vertex Intersection (C, P) (see Fig. 2(a))

vertex-edgeInterior Intersection (C, P) (see Fig. 2(b))

vertex-triangleInterior Intersection (C) (see Fig. 2(c))
edgeInterior-edgeInteriorCross Intersection (C) (Fig. 2(d))

Line intersection

edge-edgeCollinear Intersection (C, P) (see Fig. 3(a))
edge-triangleInterior Intersection (C) (see Fig. 3(b))
triangleInterior-triangleInterior Intersection (C) (Fig. 3(c))

Area Intersection

vertex-triangleInterior Intersection (P) (see Fig. 4, Fig. 5(a, b, d))
edgeInterior-edgeInterorCross Intersection (P) (Fig. 4, Fig. 5(a, b, c))
edge-triangleInterior Intersection (P) (see Fig. 5(d))
triangleInterior-triangleInterior Intersection (P) (see Fig. 4, Fig. 5(a, b, c, d))

It is possible that two triangles cross intersect in a line segment even when a triangle is on one side of the other triangle. In that case, it may be desirable to know which side of the other triangle is occupied. In Fig. 3(b), the triangle PQR (except QR which is in ABC) is on the positive side of triangle ABC. So PQR does not intersect the interior of object of triangle ABC. We will use this concept in Section VI. Section VII concludes, followed by references in Section VIII.

It should be noted that the vertex-edge intersection encompasses vertex-vertex, vertex-edgeInterior intersection, whereas the vertex-triangle intersection encompasses vertex-vertex, vertex-edgeInterior, and vertex-triangleInterior. Thus 1D JEPD cross intersection between ABC and PQR can be one of the three possibilities: (1) collinear along edges, (2) an edge of PQR lying in the plane of triangle ABC, or (3) triangles "pierce" through each other yielding an intersection segment.

IV. The Overall Algorithm (Intersection Between Triangles)

In this section, we describe the overall structure of the triangle-triangle intersection. In Section IV.A, we develop sub-algorithms that support the main algorithm at its intermediate steps. In addition to existence or nonexistence of an intersection, this algorithm also supports other auxiliary computations, (e.g. classification of intersection and the calculation of 3D intersection points, segment or area) which are necessary for some applications.

A. Description of the Overall Algorithm

The general structure of the overall triangle-triangle intersection algorithm is presented here. The description is in Python style so that it can be easily transported to programmable code. Here is the traditional approach to the algorithm, whereas our approach is presented in Section V.

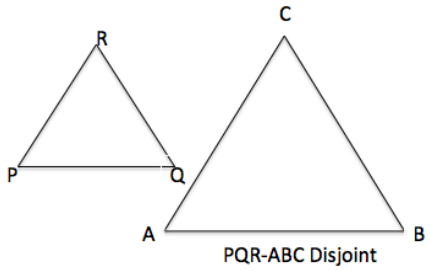

Fig. 1. Disjoint triangles: Planes supporting the triangles may be crossing or coplanar. The triangles do not have anything in common.

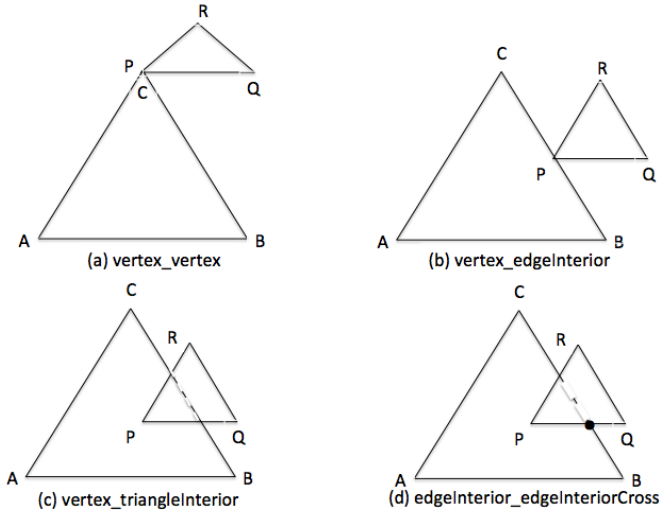

Fig. 2. Triangles intersect at a single point. The intersections between triangles ABC and PQR are JEPD (Jointly Exhaustive and Pairwise Distinct) cases of Single Point intersection between triangles. (a) vertex-vertex and (b) vertex-edgeInterior can occur in both cross and coplanar intersections. However, (c) vertex-triangleInterior and (d) edgeInterior-edgeInterior intersection point can occur in cross intersection only.

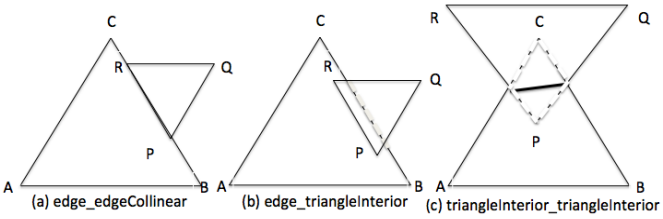

Fig. 3. Triangles intersect in a line segment. (a) edge-edgeCollinear intersection can occur in both cross and coplanar intersections. However, (b) edge-triangleInterior and (c) triangleInterior-triangleInterior intersection segment occur in cross intersection only.

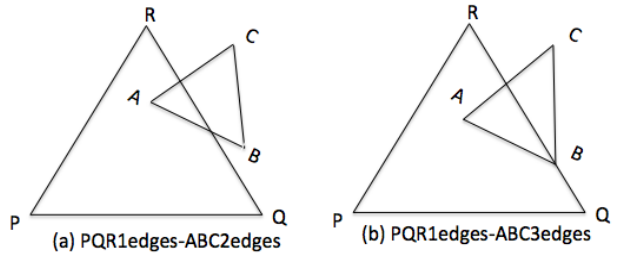

Fig. 4. Triangles intersect in an area. (a) One edge of triangle PQR and two edges AB and AC of triangle ABC intersect, vertex A is in the interior of PQR. (b) One edge of triangle PQR with three edges of ABC, and vertex A in the interior of PQR. The common area is bounded by three edges. The intersections vertex-triangleInterior, edge_triangle, edgeInterior-triangleInterior hold.

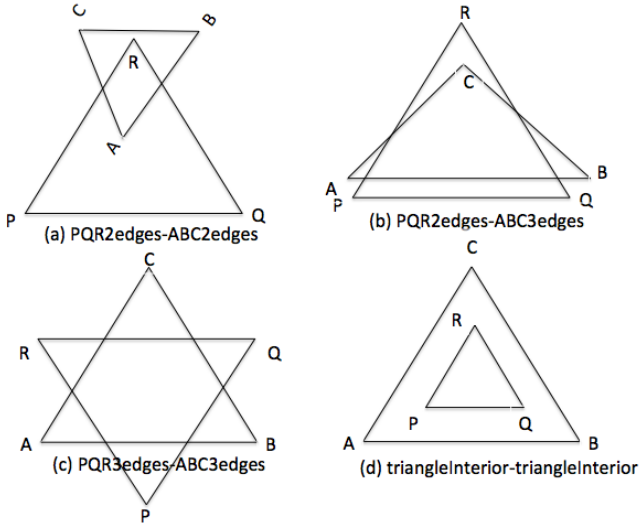

Fig. 5. Triangles intersect in an area (continued). The coplanar triangle intersections are bounded by four, five, and six edge segments. (a) Two edges of triangle PQR and two edges AB and AC of triangle ABC intersect, vertex A is in the interior of PQR, vertex R is in the interior of triangle ABC. The intersection area is bounded by four edges. (b) Two edges of triangle PQR and three edges of triangle ABC intersect; vertex C is in the interior of PQR. The intersection area is bounded by five edges. (c) Three edges of triangle PQR and three edges of triangle ABC intersect; every vertex of one triangle is outside the other triangle. The intersection area is bounded by six edges. (d) No edge of triangle PQR intersects any edge of triangle ABC; vertices P, Q, R are in the interior of triangle ABC. The intersection area is the triangle PQR.

boolean triTriInt(tr1 = ABC, tr2 = PQR)

Input: two triangles ABC and PQR

Output: Boolean value whether the triangles intersect or not.

Let ABC and PQR be two triangles. The triangles are represented with parametric vector equations where u, v are parameters for triangle ABC, and s, t are parameters for triangle PQR.

$R_1(u, v) = A + u\,U + vV$ with $0 \le u, v, u + v \le 1$
$R_2(s, t) = P + s\,S + tT$ with $0 \le s, t, s + t \le 1$

where

$U = B - A$, $V = C - A$, are directions of the edges at A;
$S = Q - P$, $T = R - P$ are the directions of edges at P.

Let $N_1 = U \times V$, $N_2 = S \times T$ be the normals to planes supporting the triangles ABC and PQR.

```
if N1xN2 ≠ 0 // planes supporting triangles are not parallel
   if triTriCrossInt (tr1, tr2) // cross intersect the triangles
      return true
   else
      return false
elseif N1xN2 = 0, // triangles planes are parallel
      if AP•N1 = 0, //the triangles are coplanar
         if triTriParInt (tr1, tr2)// implicit in Section V.
            return true
         else
            return false
      elseif AP•N1 ≠ 0, // the triangles are not coplanar,
         no Intersection
         return false
      endif
endif
/*end of algorithm*/
```

Here, we give all the supporting algorithms for implementation and classification of all special case intersections in the main algorithm. There are three broad categories for intersections of triangles: zero dimensional (single point), one-dimensional (line segment), and two dimensional (area) intersection.

A.1 Single Point Intersection (0D).

We first analyze the vertices of the triangle PQR with respect to triangle ABC to determine if a vertex P or Q or R is common to the ABC triangle and conversely.

vertex-triangleTest (X, tri = ABC)

Input: X is a vertex of one triangle and tri another triangle.

Output: boolean value determining whether X is a vertex, edgeInterior, triangleInterior point of the triangle.

To determine the relation of X $\in$ {P, Q, R} to the triangle ABC, we solve

$$A + u\,U + v\,V = X \text{ for } 0 \le u, v, u + v \le 1,$$

Rearranging the equation, we get

$$u\,U + v\,V = AX.$$

To eliminate one of the parameters u, v to solve this, we dot product the equation with vectors (UxV)xU and (UxV)xV. Let

$$\gamma = \frac{AX \times (U \times V)}{(U \times V) \bullet (U \times V)}$$

then $u = -\gamma \bullet V$ and $v = \gamma \bullet U$

if $0 \le u, v, u + v \le 1$,
  return true // X of PQR, intersects the triangle ABC.
else
  return false
/*end of algorithm*/

The vector $\frac{(U \times V)}{(U \times V) \bullet (U \times V)}$ is computed only once and used repeatedly. As a result $\gamma = \frac{AX \times (U \times V)}{(U \times V) \bullet (U \times V)}$ is calculated with one cross product, and u, v are calculated with one dot product. The parameters u, v naturally lend themselves to classification of intersections. Similarly, $\gamma' = \frac{PX \times (S \times T)}{(S \times T) \bullet (S \times T)}$.

A.2 Classification of Intersection.

In order to determine whether the vertex X of triangle PQR is a *vertex* of ABC, or on the *edge* of ABC, or an *interior point* of triangle ABC, no extra computational effort is required now. Logical tests are sufficient to establish the classification of this intersection. Since $0 \le u, v, u + v \le 1$, we can classify X relative to ABC in terms of the following predicates:

***vertex* ((u, v)):** If $(u, v) \in \{ (0, 0), (0, 1), (1, 0)\}$, then X is one of the vertices of ABC.

edgeInterior **((u, v)):** If $(u = 0, 0 < v < 1)$ or $(v = 0, 0 < u < 1)$ or $(u + v = 1, 0 < u < 1)$), then X is on an edge of ABC, excluding vertices.

triangleInterior **((u, v)):** If $(0 < u < 1$ and $0 < v < 1$ and $0 < u + v < 1)$, X is an interior point (excluding boundary) of the triangle ABC.

Similarly, as above we can classify vertex X of triangle ABC as *vertex, edgeInterior, or triangleInterior* point of triangle PQR. Single point intersection may result from cross intersection of edges as well. An edge point may be a vertex or an interior point of the edge.

A.3 The Edge-edge Single Point Intersection.

If two triangles cross intersect across an edge, the edge-to-edge intersection results in a single point. The edge-edge cross intersection algorithm is presented below.

edge_edgeCrossIntersection (edge1, edge2)

Let the two edges be AB and PQ. Then the edges are represented with equations

X = A + u U with U = B – A, $0 \le u \le 1$
X = P + s S with S = Q – P, $0 \le s \le 1$

if $U \times S \bullet AP \neq 0$, return false // non-coplanar lines
elseif $U \times S = 0$, return false // lines are parallel
else $U \times S \neq 0$, // lines cross
/* solve for u_P, s_A values for the intersection point*/

$$A + u_P U = P + s_A S$$

$u_P = S \bullet PA \times (U \times S)/(U \times S \bullet U \times S)$

$$u_P = \frac{S \bullet PA \times (U \times S)}{(U \times S) \bullet (U \times S)}$$

if $(u_P < 0)$ or $(u_P > 1)$, return false // no cross intersection,

$$s_A = \frac{U \bullet AP \times (U \times S)}{(U \times S) \bullet (U \times S)}$$

if $(s_A < 0)$ or $(s_A > 1)$,
    return false //no cross intersection,
else
    return true //there is edge-edge cross intersection.
endif
/* end of algorithm*/

A.4 Composite Classification Of Single Point Intersection.

Let A_m, P_m, be the pair of bilinear parametric coordinates of the 3D intersection points $R_1(u_m,v_m)$ and $R_2(s_m,t_m)$ with respect to triangles ABC and PQR respectively. When there is no confusion, we will refer to the points as A_m and P_m instead of 3D points $R_1(u_m,v_m)$ and $R_2(s_m,t_m)$. From vertex-triangle intersection (Section 3) we have

P_m is a vertex of PQR, and $A_m = (u_m, v_m)$, where u_m and v_m are $u_m = -\gamma \bullet V$, $v_m = \gamma \bullet U$ or A_m is a vertex of ABC, and $P_m = (s_m, t_m)$, where s_m and t_m) are $s_m = -\gamma' \bullet T$, $t_m = \gamma' \bullet S$.

From edge-edge intersection (Section B.3) we have

$A_m = (0, u_P)$ or $(u_P, 0)$ or $(u_P, 1 - u_P)$ or $(1 - u_P, u_P)$
$P_m = (0, s_A)$ or $(s_A, 0)$ or $(s_A, 1 - s_A)$ or $(1 - s_A, s_A)$

If (u_P = 0 or 1) and (s_A = 0 or 1), it is *vertex-vertex* intersection. If (u_P = 0 or 1) and not (s_A = 0 or 1), it is *vertex-edgeInterior* intersection. If not (u_P = 0 or 1) and (s_A = 0 or 1), it is *edgeInterior-vertex* intersection. If not (u_P = 0 or 1) and not (s_A = 0 or 1), it is *edgeInterior-edgeInterior* intersection. This completes the discussion of single point intersection classification and parameters for the corresponding 3D points.

B. Line Intersection (1D)

Besides edge-edge cross intersection, the edge-edge collinear intersection is a possibility, independent of crossing

or coplanar triangles. In this section, we discuss algorithms that result in a segment (1D) intersection; see Fig. 3.

B.1 Intersection Algorithm And Parametric Coordinates.

Here we derive an edge-edgeCollinear intersection algorithm. This algorithm is seamlessly applicable to both cross-intersecting and coplanar triangles. The following algorithm implements intersection of edges of the triangles ABC and PQR.

boolean edge-edgeCollinearTest (edge1, edge2)

input: two line segments

output: true if the segments have a common intersection, else false. First we compute the linear parameter coordinates u_P, u_Q, s_A, s_B for intersection of X = A + u (B – A), for X = P, Q and X = P + s (Q – P), for X = A, B. Similarly, we can compute the intersection of other edges of triangle ABC with any edge of triangle PQR. Then we update the parameters for the common segment. This algorithm is standard, straightforward and is omitted for the sake of limited space.

B.2 Classification of Edge-edge Intersection

Now we have the linear coordinates for intersection points u_P, u_Q and s_A, s_B. We map the linear parameters for intersection points to bilinear parameter coordinates (u, v) and (s, t). If u_P, u_Q are known along an edge and the edge is AB, let $u_m = u_P$, $u_M = u_Q$, $v_m = 0$, $v_M = 0$;

Similarly for AC, let $v_m = u_P$, $v_M = u_Q$, $u_m = 0$, $u_M = 0$; and for BC, let $u_m = u_P$, $u_M = u_Q$, $v_m = 1 – u_P$, $v_M = 1 – u_Q$;

Thus ABC triangle bilinear coordinates for the intersection points are:

$$A_m=(u_m, v_m),\ A_M=(u_M, v_M)$$

where $v_m = v_M = 0$ or $u_m=u_M =0$ or $u_m+v_m=u_M + v_M = 1$.

Similarly for the triangle PQR, the linear coordinates s_A, s_B of intersection translate into bilinear coordinates

$$P_m = (s_m, t_m),\ P_M = (s_M, t_M)$$

where $t_m = t_M = 0$ or $s_m = s_M = 0$ or $s_m+t_m = s_M+t_M = 1$.

Now we have the bilinear parametric coordinates u, v, s, t for the intersection segment. The common 3D segment is denoted by $[R_1(A_m), R_1(A_M)]$ which is $[R_2(P_m), R_2(P_M)]$ or $[R_2(P_M), R_2(P_m)]$. It is possible that the intersection segment is equal to both edges, or it overlaps both edges, or it is entirely contained in one edge. Since the intersection is a part of the edges, it cannot properly contain any edge.

B.3. Composite Classification of Line Intersection.

For collinear edge intersection A_m, A_M are normally distinct and similarly P_m, P_M may be distinct. Though the intersection segment is given by $[R_1(A_m), R_1(A_M)] = [R_2(P_m), R_2(P_M))$ or $[R_1(A_m), R_1(A_M)] = [R_2(P_m), R_2(P_M)]$, it is not necessary that parameter coordinates $[A_m, A_M] = [P_m, P_M]$ or $[A_m, A_M] = [P_M, P_m]$. The predicate for edge-edge collinear intersection segment becomes:

*edge-edgeCollinear (*edge1, edge2*) = edge* ($[A_m, A_M]$) and *edge* ($[P_m, P_M]$) and $[R_1(A_m), R_1(A_M)] == [R_2(P_m), R_2(P_M)]$ or $[R_1(A_m), R_1(A_M)] == [R_2(P_M), R_2(P_m)]$

Also it may be noted that for a cross intersection triangle, an *edge-triangleInterior* intersection may result in a segment intersection (Fig. 3(b)). For cross intersecting planes we have (cf. 3.A for vertex to triangle intersection and [7]) .

*edge-triangle (*edge, triangle*) = edge* ($[A_m, A_M]$) and *triangle* ($[P_m, P_M]$) and $[R_1(A_m), R_1(A_M)] == [R_2(P_m), R_2(P_M)]$ or $[R_1(A_m), R_1(A_M)] == [R_2(P_M), R_2(P_m)]$

This completes the discussion of segment intersection (1D), classification, 3D points for both cross and coplanar triangle intersections.

V. Area Intersection

For coplanar triangles, there may be no intersection (Fig. 1), a single point (Fig. 2(a, b)), a segment (Fig. 3(a)) or an area (Fig. 4, Fig. 5(a, b, c)), including one triangle contained in another, (Fig. 5(d)). An area can result from two edges of one triangle and one, two, or three edges of another triangle, or three edges from both triangles creating a star shaped figure. The resulting area is bounded by 3, 4, 5, or 6 edges. All other configurations are homeomorphic to the figures presented in this paper. For qualitative spatial reasoning, in some cases (when the knowledge of cross intersection is insufficient), we resort to coplanar intersection to distinguish the externally or tangentially connected objects.

A. General Purpose Algorithm

If a vertex of PQR is in the interior of ABC (or the converse is true), then an area intersection occurs, (Fig. 4(a, b), Fig. 5(a, b, d)). If no two edges intersect and *vertex_triangleInterior (*vertex, triangle = tr2) for every vertex of a triangle tr1, then the triangle tr1 is contained in tr2 and conversely. If no *edge-edge* intersection takes place and no vertex of one triangle is inside the other triangle (or the converse is true), then they are disjoint.

Although this algorithm may look simple, it is a new approach compared to previous approaches cited in the background section. The existing methods may use alternate edge-oriented techniques to determine the area of intersection; however, those will be limited [11]. Our algorithm is more comprehensive and analytically rigorous; it is implicitly capable of handling any specific type of intersection simultaneously, which may be a single point, a segment or an area.

THE ALGORITHM: A NOVEL APPROACH

boolean triTriIntersection (tr1 = ABC, tr2 = PQR)

The triangles ABC and PQR are

$X = A + u\,U + v\,V$ with $U = B - A$, $V = C - A$, $0 \le u, v, u + v \le 1$

$X = P + s\,S + t\,T$ with $S = Q - P$, $T = R - P$, $0 \le s, t, s + t \le 1$

The general set up for detecting intersections is to solve the equation

$$A + u\,U + v\,V = P + s\,S + t\,T$$

for u, v, s, t. If a solution exists satisfying the constraints $0 \le u, v, u + v, s, t, s + t \le 1$, then there is an intersection, else there is no intersection.

Rearranging the equation, we have

$$u\,U + v\,V = AP + s\,S + t\,T \qquad (1)$$

For simplicity in solving (1), we use the following notation.

Let α, β, γ be vectors and δ be a positive real number. Then for triangle ABC, let $AP = P - A$ be a vector, $\delta = (U \times V) \bullet (U \times V)$,

$$\alpha = \frac{S \times (U \times V)}{\delta}, \beta = \frac{T \times (U \times V)}{\delta}, \gamma = \frac{AP \times (U \times V)}{\delta}$$

Similarly, let α', β', γ' be vectors and d' be a positive real number. Then for triangle PQR, let

$PA = A - P$ be a vector, $\delta' = (S \times T) \bullet (S \times T)$

$$\alpha' = \frac{U \times (S \times T)}{\delta}, \beta' = \frac{V \times (S \times T)}{\delta}, \gamma' = \frac{PA \times (S \times T)}{\delta}.$$

For intersection between triangles ABC and PQR, on dotting equation (1) with $(U \times V) \times U$ and $(U \times V) \times V$, we quickly get

$u = -(\gamma \bullet V + s\,\alpha \bullet V + t\,\beta \bullet V)$

$v = \gamma \bullet U + s\,\alpha \bullet U + t\,\beta \bullet U$

Adding the two equations,

$$u + v = \gamma \bullet (U - V) + s\,\alpha \bullet (U - V) + t\,\beta \bullet (U - V)$$

In order that $0 \le u, v, u + v \le 1$, we get the following inequalities for possible range of values for s and t

(a) $-\gamma \bullet U \le \alpha \bullet U\, s + \beta \bullet U\, t \le 1 - \gamma \bullet U$

(b) $-1 - \gamma \bullet V \le \alpha \bullet V\, s + \beta \bullet V\, t \le -\gamma \bullet V$

(c) $-\gamma \bullet (U - V) \le \alpha \bullet (U - V)\, s + \beta \bullet (U - V)\, t \le 1 - \gamma \bullet (U - V)$

These linear inequalities (a) – (c) are of the form

$$m \le ax + by \le n$$

The solution to this system of inequalities is derived at the end of this section. We apply the results of the algorithms here in solving (a) – (c).

If we solve_x $(-\gamma \bullet U, \alpha \bullet U, \beta \bullet U, 1 - \gamma \bullet U, -\gamma \bullet V, \alpha \bullet V, \beta \bullet V, 1 - \gamma \bullet V, x_m, x_M)$

$$s_m = \max(0, x_m),\ s_M = \min(1, x_M)$$

If we solve_x $(-\gamma \bullet U, \alpha \bullet U, \beta \bullet U, 1 - \gamma \bullet U, -\gamma \bullet (U - V), \alpha \bullet (U - V), \beta \bullet (U - V), 1 - \gamma \bullet (U - V), x_m, x_M)$

$$s_m = \max(s_m, x_m),\ s_M = \min(x_M, s_M)$$

If we solve_x $(-1 - \gamma \bullet V, \alpha \bullet V, \beta \bullet V, -\gamma \bullet V, -\gamma \bullet (U - V), \alpha \bullet (U - V), \beta \bullet (U - V), 1 - \gamma \bullet (U - V), x_m, x_M)$

$$s_m = \max(s_m, x_m),\ s_M = \min(x_M, s_M)$$

```
if s_m > s_M
  return false
else
  t_M = 0; t_m = 1
  for s∈[s_m, s_M] // we solve the inequalities for t
    if solve_y (– γ•U, α•U, β•U, 1 – γ•U, – γ•V, □ α•V,
      β•V, 1 – γ•V, s, y_m, y_M)
      t_m (s) = max (0, y_m), t_M (s) = min (1, y_M),
      t_m = min (t_m (s), t_m), t_M = max (t_M (s), t_M) // extent of
          overall t values
      if t_m (s) > t_M (s)
        Return false
      else
        t_m (s) ≤ t ≤ t_M (s)
        return true
/* end of algorithm */
```

We first solved the three inequalities pairwise for a range of values for s, so that $s_m \le s \le s_M$ holds good simultaneously with three inequalities. Then from this range of s values, we solved for t as a function of s such that $t_m(s) \le s \le t_M(s)$, and overall $t_m \le t_M$. If it succeeds, it ensures that there is a solution. Similarly, we determine for u-parameter and v-parameter values in terms of u to obtain the area enclosed by the two triangles. This algorithm detects whether coplanar triangles intersect, and we classify the intersection as in Section V.B. Here we describe the two algorithms we applied in the general-purpose algorithm. An auxiliary algorithm solves inequalities of the form

$$m \le ax + by \le n, \text{ and}$$
$$M \le Ax + By \le N$$

The brute force method for solving these inequalities may lead to an erroneous solution as shown in the following example. The general elimination of variables principle that works well for equations does not directly translate into solving inequalities. Such approach gives an inconsistent solution to the two inequalities

(a) $-1 \le x + y \le 1$ and
(b) $-1 \le x - y \le 1$

Since $-1 \le x - y \le 1$ is equivalent to $-1 \le -x + y \le 1$, adding and subtracting the two inequalities (a) and (b), yields an inaccurate answer $-1 \le x \le 1$, and $-1 \le y \le 1$ which is the area enclosed by dotted boundary in Fig. 6. But the accurate solution is in the shaded area in Fig. 6, which is $|x| \le 1$, and $|y| \le (1 - |x|)$.

Thus to accurately solve these two inequalities $-1 \le x + y \le 1$ and $-1 \le x - y \le 1$, we first solve these for one variable x, then use this variable value to solve for the other variable y as $-(1 - |x|) \le y \le (1 - |x|)$.

First, we solve two most general inequalities

$$m \le ax + by \le n \quad (1)$$
$$M \le Ax + By \le N \quad (2)$$

The following algorithm determines x_m, x_M such that for each x in $[x_m, x_M]$, the inequalities hold.

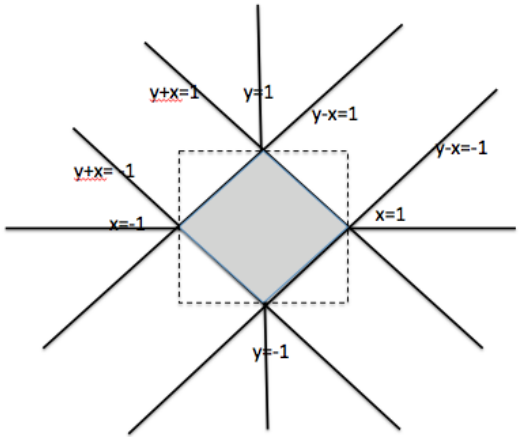

Fig. 6. Solution to a pair of inequalities: $-1 \le x + y \le 1$ and $-1 \le x - y \le 1$. Using brute force method of elimination of variables yields the area enclosed by the dotted boundary, but the accurate solution is enclosed by the shaded area.

boolean solve_x (m, a, b, n, M, A, B, N, x_m, x_M)

If a solution is found, it returns true, else it returns false. First assume b and B are non-negative. If not, multiply them by –1 to make them non-negative. Multiplying (1) by B and (2) by b, subtraction leads to

$$(mB - Mb) \le (aB - Ab)x \le (nB - Nb),$$

which yields the range $[\mathbf{x_m, x_M}]$ for x values in addition to true or false value for the algorithm.

Now once x_m, x_M have been determined, for each x in $[x_m, x_M]$ in the inequalities, we determine the range $[y_m(x), y_M(x)]$ for y. That is, after the range $[x_m, x_M]$ is determined, only then for each x in $[x_m, x_M]$, the range for y is determined; in other words, y is a function of x.

boolean solve_y (m, a, b, n, M, A, B, N, x, y_m, y_M)

Given that $x_m \le x \le x_M$ are known, it solves the inequalities for y_m, y_M . In the process, it may update the values of x_m, x_M as needed.

If a solution is found, it returns true else it returns false. Now for $x_m \le x \le x_M$, the inequalities become

$$m - ax \le by \le n - ax \text{ and}$$
$$M - Ax \le By \le N - Ax.$$

These inequalities give the range $[y_m(x), y_M(x)]$ of values for y as function of x.

This completes the general-purpose algorithm discussion for determining the triangle-triangle intersection algorithm completely.

B. Composite classification for area intersection

In this section, we summarize the algorithms in Section V.A. The equations of the triangles ABC and PQR are

$R_1(u, v) = A + u\,U + v\,V$,
where $U = B - A$, $V = C - A$, $0 \le u, v, u + v \le 1$

$R_2(s, t) = P + s\,S + t\,T$,
where $S = Q - P$, $T = R - P$, $0 \le s, t, s + t \le 1$

These equations are independent of whether they are supported by crossing planes or coplanar planes. The cross-intersecting triangles discussion is well researched, see Section II. Here we consider the general case, including crossing or coplanar triangles. In this case, the intersection may be an area in addition to a possible single point and a line segment. We first determined $[s_m, s_M]$ the range of s values, then used the range on s to solve for $[t_m(s), t_M(s)]$, the range of t. If such a solution exists, it is ensured that the two triangles intersect, which is sufficient for some qualitative spatial reasoning applications. The uv values can be similarly derived for the triangle ABC (e.g., first u_m, u_M then $v_m(u)$, $v_M(u)$). This algorithm may be used with any application (e.g., qualitative spatial reasoning, surface modeling, image processing etc.).

As described in Section III, an intersection can arise from crossing or coplanar triangles. For example, vertex-vertex or edge-edge intersection can occur regardless of triangles being coplanar or crossing. The algorithm determines whether intersection exists or not (i.e., it returns true or false). If true, the parameter coordinates of intersection are readily available. We can derive all the auxiliary information from the parametric coordinates; only logical tests are sufficient for classification of the intersections. It is not the intent of this algorithm to determine whether the triangles are crossing or coplanar.

This can be quickly determined as follows: if $U \times V \bullet S \times T \ne 0$, then triangles cross, else triangle planes are parallel. If $AP \bullet U \times V = 0$ or $AP \bullet S \times T = 0$, then the triangles are coplanar. The bilinear parameter coordinates are denoted by $A_m = (u_m, v_m)$, $A_M = (u_M, v_M)$, $P_m = (s_m, t_m)$, $P_M = (s_M, t_M)$. The intersection points can be differentiated as follows.

If the algorithm returns false,
No Intersection
Elseif ($A_m = A_M$) or ($P_m = P_M$)
Single Point Intersection
Elseif ($s_m = s_M$ or $t_m = t_M$ or $u_m = u_M$ or $v_m = v_M$)
Line segment intersection common to two triangles
Else
Area Intersection common to two triangles.

This will implicitly cover the case when a triangle is inside the other triangle as well. If triangles do not intersect, then the triangles are declared *disjoint*. This completes the discussion of overall intersection between triangles.

VI. Application to Qualitative Spatial Reasoning

Qualitative Spatial Reasoning relies on intersections between objects whose boundaries are triangulated. The spatial relations are determined by the 9-Intersection/4-Intersection model [9, 10]. That is, for any pair of objects A and B, the interior-interior intersection predicate, IntInt(A, B), has true or false value depending on whether the interior of A and the interior of B intersect without regard to precise intersection. Similarly IntBnd(A, B) represents the truth value for the intersection of the interior of A and the boundary of B, and BndBnd(A, B) represents the predicate for the intersection of the boundaries of A and B. These four qualitative spatial reasoning predicates are sufficient to define the RCC8 spatial relations (see Table 1).

In the application VRCC-3D+, the boundary of an object is already triangulated; that is, we will need to intersect pairs of only triangles. To reduce the computational complexity, the algorithm uses axis aligned bounding boxes (AABB) to determine the closest triangles that may possibly intersect. For example, for objects A and B, if bounding boxes for triangles of A are disjoint from bounding boxes for triangles of B, either A is contained in B (IntInt, BndInt is true) or B is contained in A (IntInt, IntBnd is true) or A is disjoint from B. The test for such containment of objects can be designed by casting an infinite ray through the centroid of A. If the ray intersects B an odd number of times, then B is contained in A. Similarly, the test can be made if A is contained in B. If A is not contained in B and B is not contained in A, then A and B are disjoint (i.e., IntInt(A, B), IntBnd(A, B), BndInt(A, B), and BndBnd(A, B) are all false).

If the triangles cross intersect (e.g., *triangleInterior–triangleInterior* is true), then IntInt, IntBnd, BndInt, BndBnd will be true. However if the triangles are coplanar and intersect, only BndBnd(A, B) is true and IntInt(A, B), IntBnd(A, B), BndInt(A, B) are false for the objects; otherwise, BndBnd(A, B) is also false.

It is possible that two triangles cross intersect in a line segment even when a triangle is on one side of the other triangle, so *edgeInterior–triangleInterior* is true. In that case, it may be desirable to know which side of the other triangle is occupied. In Fig. 3(b), the triangle PQR is on the positive side of triangle ABC. For example, if triangle1 of object A cross intersects the negative side of triangle2 of object B, then BndInt(A, B) is true.

Table 2 enumerates the outcome for triangle-triangle intersection with respect to 3D objects. This is a characterization of the intersection predicates, which subsequently can be used to resolve the eight RCC8 relations. Here we assume all normals are oriented towards the outside of the object. Each characterization in Table 2 describes when the associated predicate is true. If the truth test fails, then other triangles need to be tested. If no pair of triangles results in a true value, then the result is false.

TABLE I.
RCC8 relations and intersection predicates, only shaded entries are necessary.

RCC8	IntInt	BndBnd	IntBnd	BndInt
DC	F	F	F	F
EC	F	T	F	F
PO	T	T	T	T
EQ	T	T	F	F
TPP	T	T	F	T
NTPP	T	F	F	T
TPPc	T	T	T	F
NTPPc	T	F	T	F

TABLE II.
Characterization of intersection predicates

IntInt	At least one pair of triangles cross intersects (triangleInterior-triangleInterior) Or an object is contained in the other.
BndBnd	At least one pair of triangles (cross or coplanar) intersects.
BndInt	At least one pair tr1 and tr2 intersect, at least one vertex of tr1 is on the negative side of triangles of object 2. Or object 1 is contained inside object2, i.e. every vertex of object1 is on the negative side of triangles of object 2.
IntBnd	At least one pair tr1 and tr2 intersect, at least one vertex of tr2 is on the negative side of triangles of object 1. Or object 2 is contained inside object1, i.e. every vertex of object2 is on the negative side of triangles of object 1.

This characterizes the intersection predicates, which help in resolving the RCC8 relations.

VII. Conclusion

For the 9-Intersection model used in qualitative spatial reasoning, triangle-triangle intersection plays a prominent role. Herein we presented a complete framework for determining and characterizing the intersection of geometric objects. In contrast to other algorithms, our approach is a general technique to detect any type of intersection. It creates classifications by applying logical tests rather than computational arithmetic tests.

Thus, our algorithm not only detects whether or not an intersection exists, but also classifies intersections as a single

point, a line segment, or an area. The algorithm provides more information than required by spatial reasoning systems. Consequently, we hope the new ideas and additional information including classification of 3D intersection presented herein will be useful in other related applications.

REFERENCES

[1] M. J. Egenhofer, R.G. Golledge, *Spatial and Temporal Reasoning in Geographic Information Systems*, Oxford University Press, USA, 1998.

[2] E.G. Houghton, Emnett R.F., Factor J.D. and Sabharwal C.L., "Implementation of A Divide and Conquer Method for Surface Intersections," *Computer Aided Geometric Design*, Vol. 2, pp. 173–183, 1985.

[3] Oren Tropp, Ayellet Tal, Ilan Shimshoni. "A fast triangle to triangle intersection test for collision detection," *Computer Animation and Virtual Worlds*, Vol. 17 (50), pp. 527–535, 2006.

[4] G. Caumon, Collon-Drouaillet P, Le Carlier de Veslud C, Viseur S, Sausse J. "Surface-based 3D modeling of geological structures," *Math. Geosci*. 41:927–945, 2009.

[5] A.H. Elsheikh, M. Elsheikh, "A reliable triangular mesh intersection algorithm and its application in geological modeling," *Engineering with Computers*, pp. 1–15, 2012.

[6] N. Eloe, J. Leopold, C. Sabharwal, and Z. Yin, "Efficient Computation of Boundary Intersection and Error Tolerance in VRCC-3D+", *Proceedings of the 18h International Conference on Distributed Multimedia Systems (DMS'12)*, Miami, FL, Aug. 9–11, 2012, pp. 67–70, 2012.

[7] C.L. Sabharwal, J.L. Leopold, "A Fast Intersection Detection Algorithm For Qualitative Spatial Reasoning", *Proceedings of the 19h International Conference on Distributed Multimedia Systems (DMS'13)*, Brighton, UK, Aug. 8–10, 2013.

[8] D. A. Randell, Z. Cui, and A.G. Cohn. "A Spatial Logic Based on Regions and Connection," *KR*, 92, pp. 165–176, 1992.

[9] M.J. Egenhofer, R. Franzosa. "Point-Set topological Relations," *International Journal of Geographical Information Systems* 5(2), pp. 161–174, 1991.

[10] C.L. Sabharwal, J.L. Leopold. "Reducing 9-Intersection to 4-Intersection for identifying relations in region connection calculus," *24th International Conference on Computer Applications in Industry and Engineering*, pp. 118–123, 2011.

[11] P. Guigue, O. Devillers. "Fast and robust triangle-triangle overlap test using orientation predicates." *Journal of GraphicsTools*; 8(1): pp. 25–42, 2003.

[12] M. Held. "ERIT, A collection of efficient and reliable intersection tests," *Journal of Graphics Tools*; 2(4): pp. 25–44, 1997.

[13] T. Möller "A fast triangle-triangle intersection test," *Journal of Graphics Tools*, 1997; 2(2): 25–30.

[14] B. Didier, "An Efficient Ray–Polygon Intersection," Andrew S. Glassner, ed. *Graphics Gems*, Academic Press, pp. 390–393, 1990.

[15] C.L. Sabharwal, "Survey of implementations of cross intersection between triangular surfaces," *MDC Report Q0909* (Now Boeing at St. Louis, MO, USA), 1987.

ST Algorithm for Medical Diagnostic Reasoning

Irosh Fernando and Frans A. Henskens

Abstract—**The authors have previously described an approach for medical diagnostic reasoning based on the ST (Select and Test) model introduced by Ramoni and Stefanelli et al. This paper extends the previous approach by introducing the required algorithm for medical expert system development. The algorithm involves a bottom-up and recursive process using logical inferences, abduction, deduction, and induction. Pseudocode for the algorithm, and the data structures involved, are described, and the algorithm's implementation using a small sample knowledgebase and programmed in Java is included in appendixes. Implementation of a successful expert system is a challenging process; development of the necessary algorithm for its inference engine, and definition of a knowledgebase structure that models expert diagnostic reasoning and knowledge, only fulfils the initial step. Challenges associated with the remaining steps of the development process can be identified and dealt with using the CLAP software process model.**

Index Terms—**Medical diagnostic reasoning, medical expert systems, ST model.**

I. Introduction

REALISATION of medical expert systems has been one of the earliest goals of the AI community. Unfortunately, attempts by major projects such as INTERNIST-I and CADUCEUS have not been successful [1].

One of the reasons for this failure can be understood in relation to the lack of models that capture the depth and complexity of expert medical diagnostic reasoning. Models previously proposed for medical diagnostic reasoning include: scheme-inductive reasoning [2]; hypothetico-deductive reasoning [3]; backward and forward reasoning [4]; pattern recognition [5]; Parsimonious Covering Theory [6]; Information Processing Approach [7]; Process Model for diagnostic reasoning [8]; Certainty Factor model [9]; models based on Bayes Theorem [10-12]; and models based on Fuzzy logic [13-15]. The authors have previously described the limitations of some of these approaches, and proposed an approximate reasoning model for medical diagnostic reasoning [16]. This previously proposed model was based on the epistemological framework (also known as Select and Test (ST) model) for medical diagnostic reasoning proposed by Ramoni and Stefanelli et al. [17].

This paper complements the authors' previous approach by introducing the required algorithm for diagnostic inference. The previously proposed reasoning model requires of at least three layers of knowledgebase entities, namely diagnoses, symptoms and symptom attributes, together with mathematical functions to quantify those entities. In order to improve readability, the algorithm described in this paper has been deliberately simplified by restricting its application to the first two layers only. Extension of the algorithm to incorporate the full model is explained in the Discussion section of this paper.

The remainder of the paper begins with an introduction to the ST Model followed by formalisation of the knowledgebase as a graph consisting of symptoms and diagnoses. Then, the algorithm's pseudocode and the data structures, and its implementation, are described using a sample knowledgebase. Before the paper is concluded, other challenges that are faced in developing successful medical expert systems are briefly outlined, and the CLAP software process model [18] is described as a framework for addressing these challenges in a systematic manner.

II. Select And Test (ST) Model

The ST Model describes a cyclical process (Fig. 1), which uses the logical inferences, abduction, deduction, and induction that were described by Charles Peirce [19]. Usually, diagnostic reasoning in clinical contexts begins when a patient reports a symptom or symptoms to their clinician. Whilst these symptoms are well-defined entities in the clinician's mind, patients may use various descriptive terms to describe their symptoms. For example, a patient may use the descriptive term 'a dark cloud over me' to describe the symptom 'low mood'.

The process of mapping these descriptive terms understood by patients onto well-defined symptom entities used in the knowledgebase is known as abstraction. The next step, known as abduction, involves determining all likely diagnoses related to the reported symptoms. Then, for each likely diagnosis, it is necessary to determine if the patient is experiencing other expected symptoms. This is known as deduction. These three steps repeat cyclically until all the required symptoms and diagnoses have been explored. Once this cycle is ended, the final step, induction, occurs.

Manuscript received August 6, 2013; accepted for publication on September 30, 2013.

D. A. I. P. Fernando is with the School of Electrical Engineering and Computer Science, University of Newcastle, NSW 2308, Australia (phone: +61 423 281 664; e-mail: irosh.fernando@uon.edu.au).

F. A. Henskens is with the School of Electrical Engineering and Computer Science, University of Newcastle, NSW 2308, Australia (e-mail: frans.henskens@newcastle.edu.au).

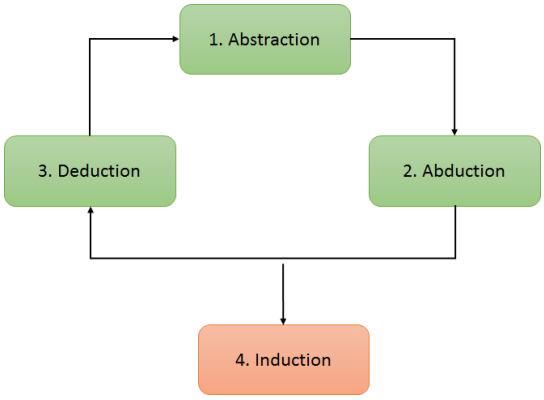

Fig. 1. The ST Model.

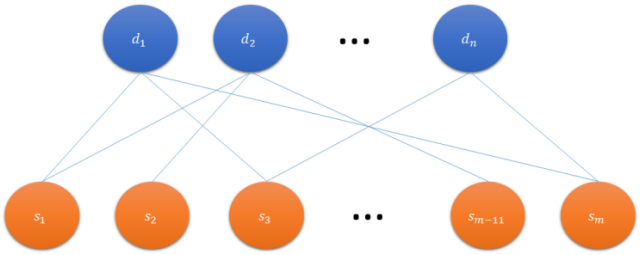

Fig. 2. Simplified knowledgebase representing diagnoses and symptoms only.

Induction involves matching the elicited symptoms with the expected symptoms for each likely diagnosis, thus determining whether the patient is suffering from any of the likely diagnoses. More details of the ST model can be found in the paper published by Ramoni and Stefanelli et al [17].

III. Formal Model For Knowledgebase

By way of formalising the process described above, let us represent all the diagnoses and symptoms in our knowledgebase as sets $D = \{d_1, d_2, \ldots, d_n\}$ and $S = \{s_1, s_2, \ldots s_m\}$ respectively. The relationship representing 'given a symptom s_i how likely is diagnosis d_j' is represented as a two-layer graph (Fig. 2), in which each arc is associated with a value θ_{ij} representing the likelihood (L) that s_i implies d_j; note $0 \leq \theta_{ij} \leq 1$. This can also be represented using the notation $L(d_j \mid s_i) = \theta_{ij}$. By way of example, in Fig. 2 the arc connecting d_1 and s_3 would have associated likelihood θ_{31}. The knowledgebase consisting of the two layers, symptoms and diagnoses, can be represented as a matrix $[\theta_{ij}]$.

IV. Select And Test Algorithm

Medical diagnostic reasoning involves two main steps:

1. search for symptoms,
2. arrive at diagnoses based on the symptoms found in the previous step.

Because of the vastness of the knowledgebase, one of the most challenging aspects of diagnostic reasoning is the symptom search process. It is therefore not uncommon that even an experienced clinician can at times miss a diagnosis because of failure to elicit a key symptom that would have provided an important clue to a diagnosis. If all the symptoms are known, arriving at a diagnosis is relatively easy computationally, depending on the commonly agreed or established diagnostic criteria used in different medical specialities. For example, in psychiatry, if all the symptoms are known, the second step involves matching the elicited symptoms with the diagnostic criteria described in a standard diagnostic manual such as DSM-V [20]. In the ST algorithm, abstraction, abduction and deduction are involved in the first step, and induction is involved in the second step.

The proposed algorithm uses five dynamic data structures, namely *symptomsFound, symptomsToBeElicited, symptomsAlreadyElicited, diagnosesToBeElicited* and *diagnosesAlreadyElicited*, which are implemented as linked lists. Also, in order to describe how the algorithm works, a static data structure *patientProfile*, which an artificial entity that encapsulates all the symptoms actually present in a patient, is used. The nature of the real world diagnostic problem is that the symptoms a patient actually has are initially unknown to the clinician. Symptom searching (the first step) in real world diagnostic reasoning can be conceptualised as an endeavour to find all the content, or at least all the clinically important symptoms, stored in *patientProfile*. In real world situations *patientProfile* is a virtual entity because it represents the patient's actual symptoms, which need to be discovered by the clinician when the patient is interviewed.

Details of the abstraction step have also been simplified in this paper. Whilst, in the real world setting, abstraction involves mapping the patient's symptom descriptions to defined knowledgebase entities, this largely mechanical matching process is omitted. Rather, it is assumed that the symptom descriptions stored in *patientProfile* correspond to the symptom descriptions used in the knowledgebase.

Implementation of abstraction in a real world application would require, for example, the patient informing symptoms by answering closed-ended questions using check boxes in a very basic human computer interface, or via an actual dialog between patient and expert system using natural language capabilities.

The data structures used in a computer-based implementation of the algorithm are described in Fig. 3, and the algorithm itself is shown in Fig. 6.

The ST Algorithm starts when a patient reports a set of initial symptoms that are stored in *symptomsFound* and *diagnosesAlreadyElicited*.

Abduction then begins, returning all the diagnoses connected to each symptom stored in *symptomsFound*. A threshold value *likelihoodThreshold* in relation to the connection strength between any symptom and related diagnosis can be used to determine which diagnoses are to be retrieved.

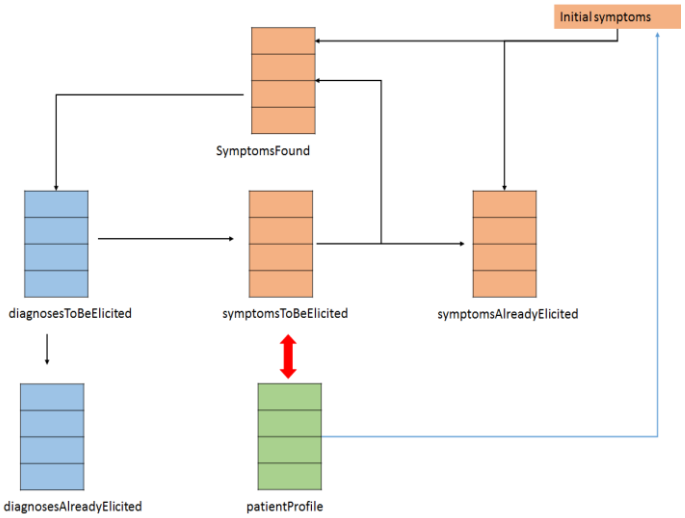

Fig. 3. Data structures used in the ST algorithm.

Accordingly, for any given symptom s_i the system will retrieve all the diagnoses d_j for which the likelihood that the symptom is caused by the diagnosis is in accordance with $\theta_{ij} >$ *likelihoodThreshold*.

These diagnoses are stored in the linked list *diagnosesToBeElicited*; before storing each diagnosis the system checks if it has already been stored in *diagnosesAlreadyElicited* because of association with a previous symptom, thus avoiding the possibility of duplicate diagnoses.

Next, deduction begins by returning all the expected symptoms connected with each diagnosis stored in *diagnosesToBeElicited*, after which the diagnosis is removed from *diagnosesToBeElicited* and transferred into *diagnosesAlreadyElicited.* All the expected symptoms that are returned for each diagnosis are transferred into *diagnosesToBeElicited* unless they are already stored in *diagnosesAlreadyElicited.*

Abstraction then commences, eliciting each symptom stored in *symptomsToBeElicited* by searching *patientProfile*. Then the elicited symptom is removed from *symptomsToBeElicited* and transferred into *symptomsAlreadyElicited* . If the elicited symptom is found in *patientProfile* it is stored in *symptomsFound*.

Finally, induction involves matching diagnostic criteria (i.e. symptoms expected for each diagnosis in *diagnosesAlreadyElicited*) with the symptoms stored in *symptomsFound*. If the diagnostic criteria are met, depending on the expected symptoms and the symptoms in *symptomsFound* then the respective diagnosis is accepted. Otherwise the respective diagnosis is excluded.

V. An Example And Its Implementation

In order to elaborate the proposed algorithm, let us consider a small knowledge base consisting of twelve symptoms and six diagnoses as described in Fig. 4. The relationships between these diagnoses and symptoms (i.e. θ_{ij} as described previously) are described in the matrix shown in Fig. 5. Java implementations of this knowledgebase and the algorithm are presented in Appendices 1 and 2, respectively.

s_1	Depressed mood
s_2	Loss of motivation
s_3	Weight loss
s_4	Fatigue
s_5	Chest discomfort
s_6	Worrying thoughts
s_7	Low self-esteem
s_8	Headache
s_9	Loss of appetite
s_{10}	Hand tremors
s_{11}	Hypertension
s_{12}	Gastrointestinal bleeding

d_1	Major Depression	0.5
d_2	Generalized Anxiety Disorder	0.2
d_3	Hyperthyroidism	0.6
d_4	Pheochromocytoma	0.7
d_5	Anaemia	0.7
d_6	Ischaemic Heart Disease	0.9

Fig. 4. Symptoms and diagnoses to be included in sample knowledgebase.

Details of the induction step are omitted in the implementation; the implementation of this step depends on the diagnostic criteria that are used to match the elicited symptoms with the expected symptoms of diagnoses. In its simplest form, the induction step can be implemented by one to one matching of the expected symptoms with the elicited symptoms. Nonetheless, depending on the diagnosis, it may not be necessary to have all the expected symptoms to make that diagnosis. In such situations, logical expression constructed using *AND* and *OR* operators can be used to formulate diagnostic rules by connecting different combination of symptoms. These diagnostic rules can be enhanced by allowing quantification of the severity of the symptoms elicited in the patient, as described elsewhere [16].

	d_1	d_2	d_3	d_4	d_5	d_6
s_1	0.9	0	0.3	0	0.3	0.3
s_2	0.9	0	0	0	0.3	0
s_3	0.6	0	0.7	0	0	0
s_4	0.6	0.7	0.6	0	0.8	0.3
s_5	0	0.6	0	0	0	0.8
s_6	0	0.9	0	0	0	0
s_7	0.6	0.4	0	0	0	0
s_8	0	0.6	0	0.5	0	0
s_9	0.7	0	0	0	0	0
s_{10}	0	0.6	0.8	0	0	0
s_{11}	0	0	0	0.9	0	0.4
s_{12}	0	0.4	0.3	0	0.6	0.6

Fig. 5. Representation of the knowledgebase as a matrix, *[θij]*.

```
1.  // Declare dynamic data structures as linked lists
2.  symptomsFound;
3.   symptomsToBeElicited;
4.  symptomsAlreadyElicited;
5. diagnosesToBeElicited;
6.  diagnosesAlreadyElicited;

7.   // Declare the threshold value for the likelihood of diagnoses
8. likelihoodThreshold;
9.  BEGIN
10.
11. Store the symptoms initially reported in symptomsFound;
12.
13.      // ABDUCTION: get all the diagnoses related to symptoms
14.      //-------------------------------------------------------------------------------
15.    FOR EACH symptom in symptomsFound DO
16.          declare diagnoses as a temporary linked list
17.          Get all the diagnoses related to symptom above the likelihoodThreshold and store in diagnoses;
18.           FOR EACH diagnosis in diagnoses DO
19.                  IF diagnosis is NOT already in (diagnosesToBeElicited OR diagnosesAlreadyElicited) THEN
20.                          Store diagnosis in diagnosesToBeElicited
21.                  END-IF
22.            END-FOR
23.
24.           // DEDUCTION: get all the symptoms related to diagnoses
25.           //-------------------------------------------------------------------------------
26.          WHILE diagnosesToBeElicited is NOT empty  DO
27.             declare symptoms as a temporary linked list
28.             Get all the symptoms related to the current diagnosis in diagnosesToBeElicited above the likelihoodThreshold and store in symptoms;
29.               FOR EACH symptom in symptoms DO
30.                    IF symptom is NOT already in ( symptomsFound OR symptomsAlreadyElicited) THEN
31.                         Store symptom in symptomsToBeElicited;
32.                  END-IF
33.               END-FOR
34.               Remove the current diagnosis from diagnosesToBeElicited and store it in diagnosesAlreadyElicited;
35.               Next diagnosis in diagnosesToBeElicited becomes the current diagnosis
36.            END-WHILE
37.
38.          // ABSTRACTION: Check if the expected symptoms in likely diagnoses are found in patient
39.          //-------------------------------------------------------------------------------------------
40.          WHILE symptomsToBeElicited is NOT empty  DO
41.               IF the current symptom in symptomsToBeElicited is found in patientProfile THEN
42.                    store the current symptoms in symptomsFound;
43.               END-IF
44.                  Store the current symptom in symptomsAlreadyElicited;
45.                 Remove the current symptom from symptomsToBeElicited;
46.                 Next symptom in symptomsToBeElicited becomes the current symptom;
47.          END-WHILE
48.
49.
50.   END-FOR EACH
51.
52.   // INDUCTION: Check if the likely diagnoses meet their diagnostic criteria
53.   //-------------------------------------------------------------------------------
54.     FOR EACH diagnosis in diagnosesAlreadyElicited DO
55.          IF the diagnostic criteria of diagnosis are met based on the symptoms stored in symptomsFound THEN
56.                   diagnosis is included;
57.          ELSE
58.                   diagnosis is excluded;
59.           END-IF
60.      END-FOR
61.
62. END
```

Fig. 6. ST algorithm.

For example, consider the diagnosis d_1 = *Major Depression*, s_1 = *Depressed Mood*, and the related symptoms s_1 = *Depressed Mood*, s_2 = *Loss of Motivation*, s_3 = *Weight Loss*, and s_7 = *Low Self Esteem*.

Suppose we have a patient who presents with the above symptoms, each with a different level of severity. Let us assume that the severity of these symptoms (i.e. quantification) corresponds to q_1, q_2, q_3 and q_7 respectively. Using threshold values t_{11}, t_{12}, t_{13} and t_{17} respectively for each of these symptoms in relation to d_1, an example of a diagnostic rule is as follows:

$$IF(q_1>t_{11}\ AND\ q_2>t_{12}\ AND\ q_3>t_{13}\ AND\ q_7>t_{17})\ THEN\ accepted(d_1) = TRUE$$

where $accepted(d_1)$ indicates whether the diagnostic criteria for d_1 is met, resulting in its acceptance (or rejection) as a diagnosis. It may require several such diagnostic rules for each diagnosis, and some of the rules may also require the logical operator OR in addition to AND. Developers may have to consult standard diagnostic manuals (for example, DSM V [20] in psychiatry) when formulating the diagnostic rules.

VI. Discussion

The knowledgebase model and algorithm presented above represent a simplified version of what it is required for effective diagnostic inference in real world settings. Nonetheless, they encapsulate the essential basic characteristics of the reasoning process. This basic structure can be extended and customised according to the characteristics of clinical knowledge in various medical subspecialties (i.e. subdomains). For example, in psychiatry, the knowledgebase may require addition of an extra layer known as *clinical phenomenon* between the symptoms and diagnoses layers [21]. Also, an extra layer of symptom attributes can be added below the symptoms layer, and each symptom can be quantified using the values associated with the related symptom's attributes using mathematical functions that approximate their relationships, as described elsewhere [16].

In addition to searching for diagnoses related to a given symptom based on likelihood, as described in the algorithm, diagnostic reasoning in the real world setting also involves searching for more critical (i.e. associated with relatively worse consequences if undetected) diagnoses even though their likelihoods seem low based on the patient's reported symptoms. The ST algorithm does an exhaustive search, and therefore can be useful in ruling out more critical diagnoses that can present with rather atypical symptoms. It is possible to enhance ST by introducing a critical value δ_j associated with each diagnosis d_j that determines the level of criticality of the diagnosis. Similarly to the *likelihoodThreshold* described previously, a threshold value *criticalityThreshold* can be used to select diagnoses for which δ_j>*criticalityThreshold*.

The next significant challenge to implementing the algorithm in a practically useful expert system is developing and maintaining a sufficiently large knowledgebase. Because of the vastness of the knowledgebase and the amount of manpower and commitment required to develop and maintain it, a sufficient database has been very difficult to achieve using traditional development methods [22]. For example, despite expending nearly 25-30 person years of work, it has still not been able to complete the knowledgebase of INTERNIST-I, an expert diagnostic system in Internal Medicine [1].

Even if the required knowledgebase were implemented, there are yet more challenges. An important challenge is engaging clinicians, who may often feel threatened by medical expert systems on the grounds they may be intended to duplicate and replace some of their skills [23]. The authors have previously discussed these challenges, and introduced a software process model known as a Collaborative and Layered Approach (CLAP) as a strategy to deal with these challenging issues [18].

The main layers and the activities within each layer of CLAP are shown in Fig. 7, and the reader is encouraged to refer to the main paper on this model for more details [18]. The form of ST algorithm introduced in this paper can be considered as the main product of the conceptual layer, which primarily deals with conceptualising the expert medical reasoning process and the knowledgebase, and then translating into a formal model. The societal layer then deals with engaging clinicians in a collaborative development process, and defining the role of the under-development expert system within the complex modern day organisational structure of healthcare services in which it will be used. Finally, the computational layer deals with software and hardware implementation of the expert system. As an important strategy to overcome the difficulty of developing and maintaining the knowledgebase, the CLAP model suggests use of an online collaborative approach [24], which can be realised due to advancement of Internet-based social networking platforms.

VII. Conclusion

Whilst acknowledging the challenges in developing successful medical expert systems, this paper introduced a simplified version of the algorithm and data structures required for implementing an inference engine and knowledgebase, based on a previously introduced diagnostic reasoning model [16]. Even though there are many diagnostic reasoning models that have been previously introduced, the authors claim that the reasoning model on which the algorithm introduced in ST this paper is designed, is more comprehensive in relation to the overall expert diagnostic reasoning process.

Furthermore, the algorithm closely models the recursive steps that are involved in real world diagnostic reasoning, using logical inferences. Because of the complexity and the space required to describe the full algorithm and its implementation, it was necessary in this paper to simplify the

algorithm and knowledgebase described. However, the paper still provides the core structure on which, the full model can be built. As a means to identifying and resolving other challenges associated with the development process, the authors suggest use of the CLAP software process model for developing medical expert systems [18].

APPENDICES

Appendix 1. Java representation of the knowledgebase

```
package diagnosticalgorithm;
import java.util.ArrayList;

/**
 * @author Irosh Fernando
 */
public class Knowledgebase {

   // Declare one-dimensional array of symptoms
    static String symptoms[]= {
      "Depressed mood",      /* 1 */
      "Loss of motivation",  /* 2 */
      "Weight loss",         /* 3 */
      "Fatigue"  ,           /* 4 */
      "Chest discomfort",    /* 5 */
      "Worrying thoughts",   /* 6 */
      "Low self-esteem",     /* 7 */
      "Headache",            /* 8 */
      "Loss of appetite",    /* 9 */
      "Hand tremors",        /* 10 */
      "Hypertension",        /* 11 */
      "Dizzinesse"           /* 12 */
     };

    // Declare one dimensional array of diagnoses
    static String diagnoses[]= {
      "Major Depression",               /* 1 */
      "Generalised Anxiety Disorder",   /* 2 */
      "Hyperthyroidism",                /* 3 */
      "Paechromocytoma",                /* 4 */
      "Anaemia",                        /* 5 */
      "Ischaemic Heart Disease",        /* 6 */
     };

  // Declare the knowledgebase as a two dimensional array

    static double diag_symp[][]={
/* symptoms index:  1,   2,   3,   4,   5,   6,   7,   8,   9,   10,  11,  12  */
/* diagnosis index: 1 */ { 0.9, 0.9, 0.6, 0.6, 0.0, 0.0, 0.6, 0.0, 0.7, 0.0, 0.0, 0.0 },
/* diagnosis index: 2 */ { 0.0, 0.0, 0.0, 0.7, 0.6, 0.9, 0.4, 0.6, 0.0, 0.6, 0.0, 0.4 },
/* diagnosis index: 3 */ { 0.3, 0.0, 0.7, 0.6, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8, 0.0, 0.3 },
/* diagnosis index: 4 */ { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.9, 0.0 },
/* diagnosis index: 5 */ { 0.0, 0.3, 0.0, 0.8, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6 },
/* diagnosis index: 6 */ { 0.3, 0.0, 0.0, 0.3, 0.8, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.6 },
};

    // return the index of a given symptom
    static public int getSymptomIndex(String symptom){
       // return -1 if not found
       int index=-1;
       for( int i=0; i<symptoms.length; i++){
          if( symptoms[i].equalsIgnoreCase(symptom))
              index=i;
       }
       return index;
    }

   // return the index of a given diagnosis
    static public int getDiagnosisIndex(String diagnosis){
      // return -1 if not found
       int index=-1;
       for( int i=0; i<diagnoses.length; i++){
          if( diagnoses[i].equalsIgnoreCase(diagnosis))
              index=i;
       }
       return index;
    }

    // return all diagnoses related a given symptom above a given threshold
    static public ArrayList getDiagnoses(String symptom, double threshold){
       ArrayList<String> diagnosesList = new ArrayList<>();
       int index= getSymptomIndex(symptom);
       for( int i=0; i< diagnoses.length; i++){
          if(diag_symp[i][index]> threshold )
             diagnosesList.add(diagnoses[i]);
       }
       return diagnosesList;
    }

     // return all symptoms related a given diagnosis above a given threshold
    static public ArrayList getSymptoms(String diagnosis, double threshold){
       ArrayList<String> symptomList = new ArrayList<>();
       int index= getDiagnosisIndex(diagnosis);
       for( int i=0; i< symptoms.length; i++){
          if(diag_symp[index][i]> threshold )
             symptomList.add(symptoms[i]);
       }
       return symptomList;
    }
}
```

Appendix 2. Java representation of the algorithm

```
package diagnosticalgorithm;
import java.util.ArrayList;
import java.util.List;

/**
* @author Irosh Fernando
 * @Date 30th of June 2013
 */
public class STAlgorithm {
   static Knowledgebase KB;
   static PatientProfiles Patient;

   static List<String> symptomsFound = new ArrayList<>();
   static List<String> symptomsToBeElicited = new ArrayList<>();

   // To store both symptoms found and not found
   static List<String> symptomsAlreadyElicited = new ArrayList<>();

   // store diagnose of which symptoms are to be explored
   static List<String> diagnosesToBeElicited = new ArrayList<>();

   // Store diagnoses of which symptoms have already been explored
   static List<String> diagnosesAlreadyElicited = new ArrayList<>();

   // set the likelihood threshold
   static double likelihoodThreshold=0.5;

   // Initialise the symptoms reported by patient at the beginning
   static private void initialise(){
      symptomsFound.add("Depressed Mood");
```

```
    symptomsAlreadyElicited.add("Depressed Mood");
    //...add more symptoms as necessary
}

// Abduction
  static private void doAbduction(){
    for(int i=0; i<symptomsFound.size();i++){
      ArrayList<String> diagList;
      diagList = KB.getDiagnoses(symptomsFound.get(i),
                  likelihoodThreshold);
      // insert each diagnosis into likelyDiagnoses if not already in
      for(int j=0; j< diagList.size();j++){
        if (!diagnosesAlreadyElicited.contains(diagList.get(j)) )
          diagnosesToBeElicited.add(diagList.get(j));
      }
      doDeduction();
      doAbstraction();
    }
  }

  // Deduction
  static private void doDeduction(){
    for(int i=0; i<diagnosesToBeElicited.size();i++){
      ArrayList<String> sympList;
      sympList = KB.getSymptoms(diagnosesToBeElicited.get(i),
                  likelihoodThreshold);
// insert each expected symptom into symptomsToBeElicited if not already in
      for(int j=0; j< sympList.size();j++){
        if (!symptomsAlreadyElicited.contains(sympList.get(j)))
          symptomsToBeElicited.add(sympList.get(j));
      }
      // store alrady found symptoms in symptomsAlreadyElicited
      diagnosesAlreadyElicited.add(diagnosesToBeElicited.get(i));
    }
    // Empty the diagnosesToBeElicited after eliciting all the diagnoses
    diagnosesToBeElicited.clear();
  }

    // Abstraction
  static private void doAbstraction(){
    for(int i=0; i<symptomsToBeElicited.size();i++){

 if(!symptomsAlreadyElicited.contains(symptomsToBeElicited.get(i)) ){
        if(Patient.symptomPresent(symptomsToBeElicited.get(i))){
          symptomsFound.add(symptomsToBeElicited.get(i));
        }
        symptomsAlreadyElicited.add(symptomsToBeElicited.get(i));
      }
    }
// Empty the symptomsToBeElicited after eliciting all the expected symptoms
  symptomsToBeElicited.clear();
}
```

REFERENCES

[1] D. A. Wolfram, "An appraisal of INTERNIST-I," *Artificial Intelligence in Medicine,* vol. 7, pp. 93-116, 1995.

[2] H. Mandin, A. Jones, W. Woloschuk, and P. Harasym, "Helping students learn to think like experts when solving clinical problems," *Academic Medicine,* vol. 72, pp. 173-179, 1997.

[3] A. S. Elstein, L. S. Shulman, and S. A. Sprafka, *Medical Problem-Solving: an Analysis of Clinical Reasoning*: Cambridge, MA: Harvard University Press 1978.

[4] E. Hunt, "Cognitive Science: Definition, Status, and Questions " *Annual Review of psychology,* vol. 40, pp. 603-629 1989.

[5] G. R. Norman, C. L. Coblentz, L. R. Brooks, and C. J. Babcook, "Expertise in visual diagnosis - a review of the literature.," *Academic Medicine,* vol. 66(suppl), pp. s78-s83, 1992.

[6] J. A. Reggia and Y. Peng, "Modeling diagnostic reasoning: a summary of parsimonious covering theory," *Computer Methods and Programs in Biomedicine,* vol. 25, pp. 125-134, 1987.

[7] P. M. Wortman, "Medical Diagnosis: An Information-Processing Approach," *Computers and Biomedical Research,* vol. 5, pp. 315-328, 1972.

[8] J. Stausberg and M. Person, "A process model of diagnostic reasoning in medicine," *International Journal of Medical Informatics,* vol. 54, pp. 9-23, 1999.

[9] E. H. Shortliffe and B. G. Buchanan, "A model of inexact reasoning in medicine," *Mathematical Biosciences,* vol. 23, pp. 351-379, 1975.

[10] S. Andreassen, F. V. Jensen, and K. G. Olesen, "Medical expert systems based on causal probabilistic networks," *International Journal of Bio-Medical Computing,* vol. 28, pp. 1-30, 1991.

[11] T. Chard and E. M. Rubenstein, "A model-based system to determine the relative value of different variables in a diagnostic system using Bayes theorem," *International Journal of Bio-Medical Computing,* vol. 24, pp. 133-142, 1989.

[12] B. S. Todd, R. Stamper, and P. Macpherson, "A probabilistic rule-based expert system," *International Journal of Bio-Medical Computing,* vol. 33, pp. 129-148, 1993.

[13] K. Boegl, K. P. Adlassnig, Y. Hayashi, T. E. Rothenfluh, and H. Leitich, "Knowledge acquisition in the fuzzy knowledge representation framework of a medical consultation system," *Artificial Intelligence in Medicine,* vol. 30, pp. 1-26, 2004.

[14] L. Godo, R. L. de Mántaras, J. Puyol-Gruart, and C. Sierra, "Renoir, Pneumon-IA and Terap-IA: three medical applications based on fuzzy logic," *Artificial Intelligence in Medicine,* vol. 21, pp. 153-162, 2001.

[15] T. Vetterlein and A. Ciabattoni, "On the (fuzzy) logical content of CADIAG-2," *Fuzzy Sets and Systems,* vol. 161, pp. 1941-1958, 2010.

[16] I. Fernando, F. Henskens, and M. Cohen, "An Approximate Reasoning Model for Medical Diagnosis," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. vol. 492, R. Lee, Ed., ed: Springer International Publishing, 2013, pp. 11-24.

[17] M. Ramoni, M. Stefanelli, L. Magnani, and G. Barosi, "An epistemological framework for medical knowledge-based systems " *IEEE Transactions on Systems, Man and Cybernetics,* vol. 22, pp. 1361-1375, 1992.

[18] I. Fernando, F. Henskens, and M. Cohen, "A Collaborative and Layered Approach (CLAP) for Medical Expert System Development: A Software Process Model," in *IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS12)*, 2012, pp. 497-502.

[19] C. S. Peirce, "Illustrations of the logic of science, sixth paper-deduction, induction, hypothesis," *The Popular Science Monthly,* vol. 1, pp. 470-482, 1878.

[20] American Psychiatric Association, *Diagnostic and Statistical Manual of Mental Disorders: DSM-5*: American Psychiatric Publishing Incorporated, 2013.

[21] I. Fernando, M. Cohen, and F. Henskens, "A systematic approach to clinical reasoning in psychiatry," *Australasian Psychiatry,* vol. 21, pp. 224-230, 2013.

[22] E. L. Kinney, "Medical Expert Systems - Who needs them ?," *CHEST,* vol. 91, pp. 3-4, 1987.

[23] A. K. Das, "Computers in Psychiatry: A Review of Past Programs and an Analysis of Historical Trends," *Psychiatric Quarterly,* vol. 73, pp. 351-365, 2002.

[24] D. Richards, "Collaborative Knowledge Engineering: Socialising Expert Systems," in *11th International Conference on Computer Supported Cooperative Work in Design*, 2007.

A Logic Programming Approach to the Conservation of Buildings Based on an Extension of the Eindhoven Classification Model

Guida Gomes, Henrique Vicente, Joaquim Macedo, Victor Alves, and José Neves

Abstract—**The identification, classification and recording of events that may lead to the deterioration of buildings are crucial for the development of appropriate repair strategies. This work presents an extension of the *Eindhoven Classification Model* to sort adverse events root causes for *Building Conservation*. *Logic Programming* was used for knowledge representation and reasoning, letting the modelling of the universe of discourse in terms of defective data, information and knowledge. Indeed, a systematization of the evolution process of the body of knowledge in terms of a new factor, the *Quality of Information* one, embedded in the *Root Cause Analysis* was accomplished, i.e., the system proposed led to a process of *Quality of Information* quantification that permit the study of the event's root causes, on time.**

Index Terms—**Building conservation, Eindhoven classification model, knowledge representation and reasoning, logic programming, quality of information.**

I. Introduction

THE use of information systems as a tool for acquisition, storage and manipulation of data represents the minimum level that may be required from the information technology. In fact, presently more than the automation of processes and the increase of the data repositories are required. The focus is placed on the ability of the information systems to be an autonomous process of evaluation, decision and learning. This configures a transversal dimension that encompasses various scientific areas.

The application of methodologies emanating from the Scientific Area of Artificial Intelligence to solve problems in the field of Civil Engineering is not new, dating from the early 90s of XX century. Since then several studies have been published where techniques like Artificial Neural Networks and Genetic Algorithms have been applied to solve some specific problems within the Civil Engineering portfolio [1]. Recently Lu et al. [2] presented an overview of the application of new methodologies developed in the field of Artificial Intelligence to Civil Engineering. Among them some should be highlighted, like Evolutionary Computation, Swarm Intelligence, Fuzzy Systems, Reasoning Based Systems and Chaos Theory.

Dukić et al. [3] present a model to facilitate the planning of maintenance activities, in order to rationalize costs through preventive interventions. The system can store the information obtained in the regular inspections and based on them, infer about possible failures and/or loss of the buildings' functional characteristics. Furthermore the database allows monitoring the behavior of the various elements of construction. Motawa and Almarshad [4] developed an integrated system for archiving information and knowledge regarding the maintenance of buildings. The proposed system aims at the understanding of the causes of building deterioration, but also acts as a decision support system regarding preventive or corrective maintenance actions. This system comprises a registration module, a database and a knowledge extraction module for the construction of a knowledge base.

However, the machinery mentioned above does not work with incomplete, unknown and/or forbidden information. In fact, for many situations that occur daily in building conservation complete information does not exist at all. Instead, the information available is insufficient or incomplete.

Undeniably the building conservation area is complex and multifaceted and various types of adverse events may occur. An adverse event may be defined as the failure of a planned action to be completed as intended or the use of a wrong plan to achieve an aim, and includes problems in practice, relationships, procedures and systems. The most effective way to prevent adverse events is to attack directly their causes. Preventing the adverse events' root causes improves significantly the conservation/maintenance of buildings. Thus, the proposed model will focus primarily on preventing the adverse events' root causes. The model planned serves as the formal foundation to an adverse event reporting and learning computational system.

II. The Computational Model

An extended version of the *Eindhoven Classification Model* (*ECM*), with the extensions and adaptations for the area of

Manuscript received on August 6, 2013; accepted for publication on September 30, 2013.

Guida Gomes is with the Department of Informatics, University of Minho, Braga, Portugal (e-mail: mguida.mgomes@gmail.com).

Henrique Vicente is with the Department of Chemistry & Évora Chemistry Centre, University of Évora, Évora, Portugal (e-mail: hvicente@uevora.pt).

Joaquim Macedo, Victor Alves, and José Neves are with the Department of Informatics, University of Minho, Braga, Portugal (e-mail: {macedo, valves, jneves}@di.uminho.pt).

conservation and maintenance of buildings and its causal tree, used to classify the adverse events' root causes in conservation / maintenance of buildings, is presented. The theoretical foundation is based on an extension to *Logic Programming*, in terms of a revision of its knowledge representation and reasoning mechanisms. The introduction of explicit negation in this universe endorsed the development of a process of quantification of the above mentioned *Quality of Information* (*QoI*) factor, embedded in the predicates extensions that make one´s system, making possible to study the event's root causes and to generate alerts and recommendations in order to improve the state of building conservation and maintenance.

A. The Eindhoven Classification Model

The *ECM* was originally developed in order to manage human error in the chemical industry [5], being then applied to other industrial arenas, such as energy production, steel industry and healthcare. The *Eindhoven Classification Model – Medical Version* consists of 20 (twenty) codes, divided into four categories frequently used in a medical environment to classify the underlying causes of the adverse events [6], and recently was extended and adapted for the specific area of imaging [7]. This approach assumes that humans are fallible and that errors are to be expected in every organization, so it is necessary to concentrate efforts on the conditions under which individuals work and try to build defenses to avert errors or to mitigate their effects. Assigning codes to the causes of each adverse event, it is a practice that is useful for tracking and trending.

The first stage to use the *ECM* based classification system is to identify the root causes that result in a specific adverse event. These root causes are subsequently classified according to the classification model. Indeed, a causal tree is built and techniques of *Root Cause Analysis* (*RCA*) are applied. Once the root causes are identified, they may be used to provide a more realistic view of how the system really works, as well as to contribute to the creation of effective and lasting solutions.

B. The Extended Eindhoven Classification Model

The *Extended Eindhoven Classification Model* (*EECM*) was adapted from the *ECM*, presented in the previous section. To apply this model to the area of conservation and building maintenance, the authors developed extensions for each category of the original model. These extensions allow fitting each category into the area of conservation and maintenance of buildings and provide a broader view of the events that may occur and the degree of complexity of this field. Thus, the classification process becomes easier and more efficient. Table I shows the five categories that make up the model, a brief description of each one of them and the respective codes, while in Table II a subset of the *EECM* codes and some examples of adverse events are present. Figure 1, in turn, depicts the *EECM* flow chart.

For instance, in the original model, the adverse events classified as "*Human behaviour – Knowledge-based errors*" (*HKK*) occur due to "*the inability of an individual to apply existing knowledge to a new situation*". In the *EECM*, this definition was extended by saying that the events classified under this category are due to "*difficulties in execution, interpretation or reporting procedures*". Some of the adverse events falling into this category are "*poorly executed procedures*, *incomplete procedures* and *procedures poorly validated*".

The causal trees taken on by the original *ECM*, set that the recognition of the event's root causes and its mental picture, is done under a hierarchical structure. On the other hand, once one has to deal with incomplete and even contradictory information, an *Extension of Logic Programming* (*ELP*) was used for knowledge representation and reasoning, in order to get a truth value in the interval [0, 1] as a measure of confidence in any qualification process susceptible to be handled by the system. Since an event may only occur due to the combination of more than one cause, and a different event may come about due to two or more causes, taken separately, in the original model *AND-gates* and *OR-gates* are used to embody these two possibilities in the causal tree.

The usual situations may also include the case where only one cause leads to the occurrence of a certain event. In any case the adverse events' origins are known, i.e., there is certainty about the events' grounds. Beyond these situations, it may happen that the causes of an event, action or decision are unknown; it may be known that certain views are the source of a given event, but it may not be sure what are the event grounds; or it is not allowed to know the origin of a given event (e.g. due to internal policies of the organization in charge of maintaining the building).

Therefore, it is proposed the use of "*unknown*" and "*forbidden*" operators, to allow for a representation of unknown values of an infinite set of values, unknown values of a given set of values, and values not allowed or forbidden. The information contained in each causal tree is then represented in *ELP* by the extensions of a predicates set, being also used as a formalism to quantify the causal tree´s *QoI* (see Section 2.4). The *QoI* allows the identification of the causes that should be taken into account, in first place, and how this hampers all the classification process.

The information obtained in this way to the *RCA* enables automatic report generation with improvements in the recommendations. Figure 2 presents the application of the *EECM* to the adverse event "*study not available*". In the source of this event there is a great diversity of reasons. It is possible that only one situation might be enough for the event to occur or, perhaps, it may be necessary a combination of several factors. The causal trees should include all possible causes and aim to be a generic representation of the problem. For a particular occurrence of the event, its causes will fall on a branch of the tree.

TABLE I.
CATEGORIES OF THE EXTENDED EINDHOVEN CLASSIFICATION MODEL FOR CONSERVATION AND MAINTENANCE OF BUILDINGS AND RESPECTIVE CODES

Category	Description	Code
Technical		
External	Technical failures beyond the control and responsibility of the organization.	TEX
Design	Failures due to the poor design of the building project.	TD
Construction	Construction faults despite a well drawn up building project.	TC
Materials	Failures due to the materials used.	TSR
Structural Response	Failures due to the structural response of the buildings.	TM
…	…	…
Environmental		
Climate	Faults relating with the climate factors that the buildings are subjected to.	EC
Geotechnical	Failures related to geotechnical aspects of the place where the buildings are implanted (soil mechanics).	EG
…	…	…
Organizational		
External	Failures at an organizational level beyond the control of the organization, such as in another department or area.	OEX
Transfer of Knowledge	Failures resulting from inadequate options that do not ensure that the knowledge is transmitted to inexperienced staff.	OK
Protocols	Failures related to the quality/availability of the internal protocols (too complex/simple, unclear, or nonexistent).	OP
Management Priorities	Internal decisions in which safety is relegated to an inferior position reflecting a conflict between productivity and safety.	OM
Culture	Failures resulting from the collective approach and/or risk behaviors.	OC
…	…	…
Human behaviour		
External	Human failures originating beyond the control of the organization, such as in another department or area.	HEX
Knowledge-Based Behavior		
Knowledge-Based Errors	The inability of an individual to apply existing knowledge to a new situation.	HKK
…	…	…
Rule-Based Behavior		
Qualifications	Incorrect fit between an individual's qualifications, training, or education and a particular task.	HRQ
Coordination	Lack of task coordination within a team in an organization (e.g., an essential task not performed because everyone thought that someone else had completed the task).	HRC
Verification	Failures in the correct and complete assessment of a situation before starting the intervention. Includes the relevant conditions of buildings and materials to be used.	HRV
Intervention	Failures that result from faulty planning of task and/or poor execution.	HRI
Monitoring	Failures during monitoring of a activity/process during or after a rehabilitation intervention.	HRM
…	…	…
Skill-Based Behavior		
Slips	Failures in the performance of a task due to the lack of fine motor skills of the technician.	HSS
…	…	…
Other		
Technicians Related Factor	Failures related to physical and/or psychic conditions of the technician that influence the task performance and are beyond the control of the organization.	TRF
…	…	…
Unclassifiable	Failures that cannot be classified in any other category.	X

TABLE II.
A SUBSET OF CODES OF THE EXTENDED EINDHOVEN CLASSIFICATION MODEL FOR CONSERVATION AND MAINTENANCE OF BUILDINGS AND SOME EXAMPLES OF POSSIBLE ADVERSE EVENTS

Code	Extension to the conservation and maintenance of buildings	Examples
TD	Difficulties in the elaboration of projects. Failures sizing.	Lack of details. Overloads not provided. Specifications of recoating improper.
TC	Difficulties in interpreting projects. Lack of inspection.	Armature badly positioned. Lack of cure or cure poorly executed. Concrete with excess of water.
HKK	Difficulties in execution, interpretation or reporting procedures.	Poorly executed procedures. Incomplete procedures. Procedures poorly validated.

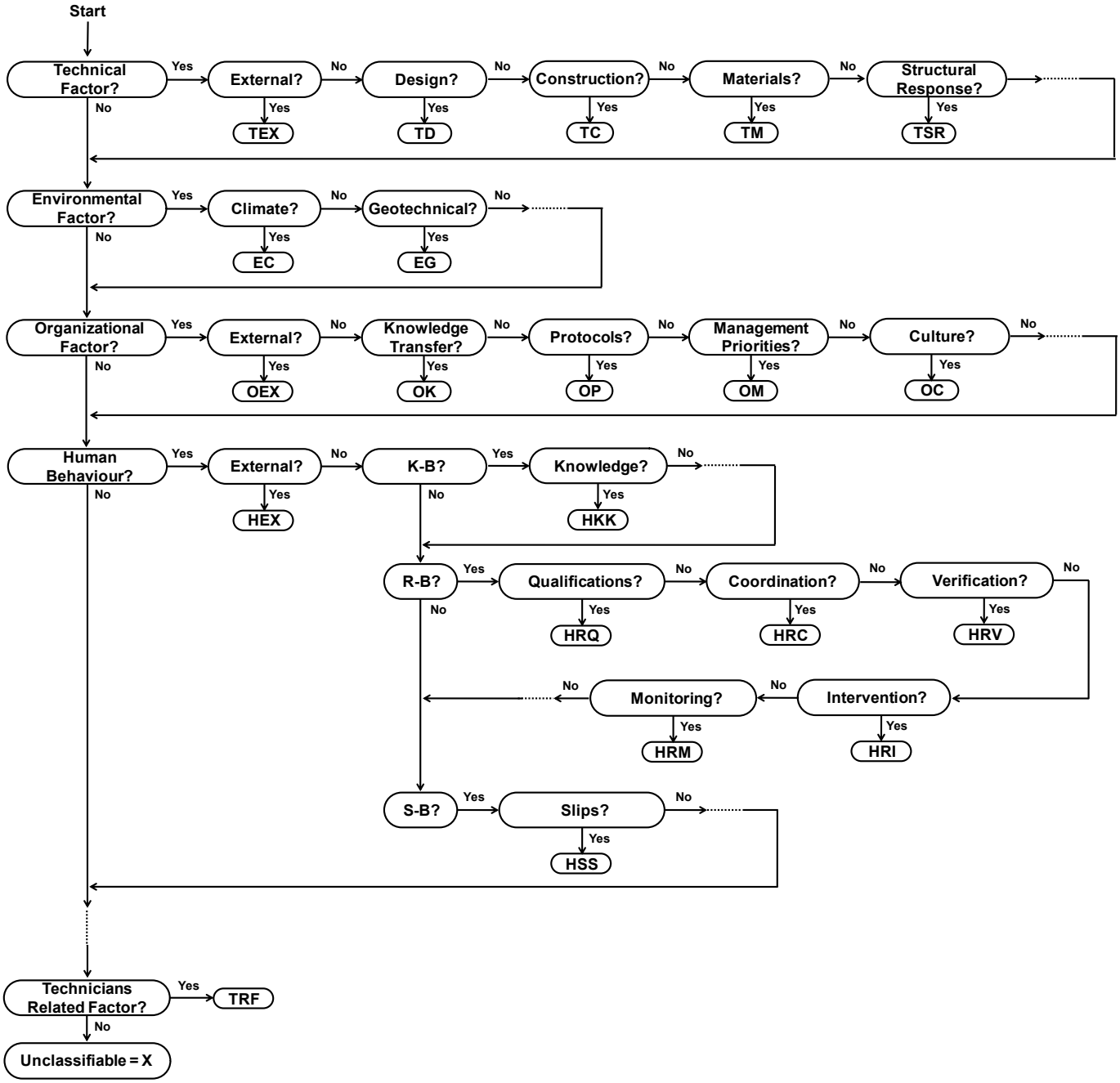

Fig. 1. The Extended Eindhoven Classification Model for Conservation and Maintenance of Buildings flow chart

C. Knowledge Representation and Reasoning

A few decades ago non-classical techniques for modelling the universe of discourse and the reasoning procedures of intelligent systems have been proposed, in addition to the classical ones [8]. Of particular interest to this work are the techniques to deal with incomplete, inconsistent, contradictory, default and forbidden information [9]. Intelligent systems require the ability to reason with incomplete information, since in the real world complete information is hard to obtain, even in the most controlled situations. The idea behind default information is the ability to make assumptions or to jump to a plausible conclusion, derived from a knowledge base in the absence of information to the contrary. The derived information is defeasible, because in light of new information the conclusion may need to be retracted, i.e., we are in the presence of non-monotonic reasoning [9], [10]. A suitable logic is needed, one that permits the representation of incomplete, inconsistent and default information and supports non-monotonic reasoning. In a classical logical theory or logic program, the proof of a theorem (here understood as a question submitted to the classification system) the outcome is a truth value, namely false (0) or true (1), i.e., {0, 1}.

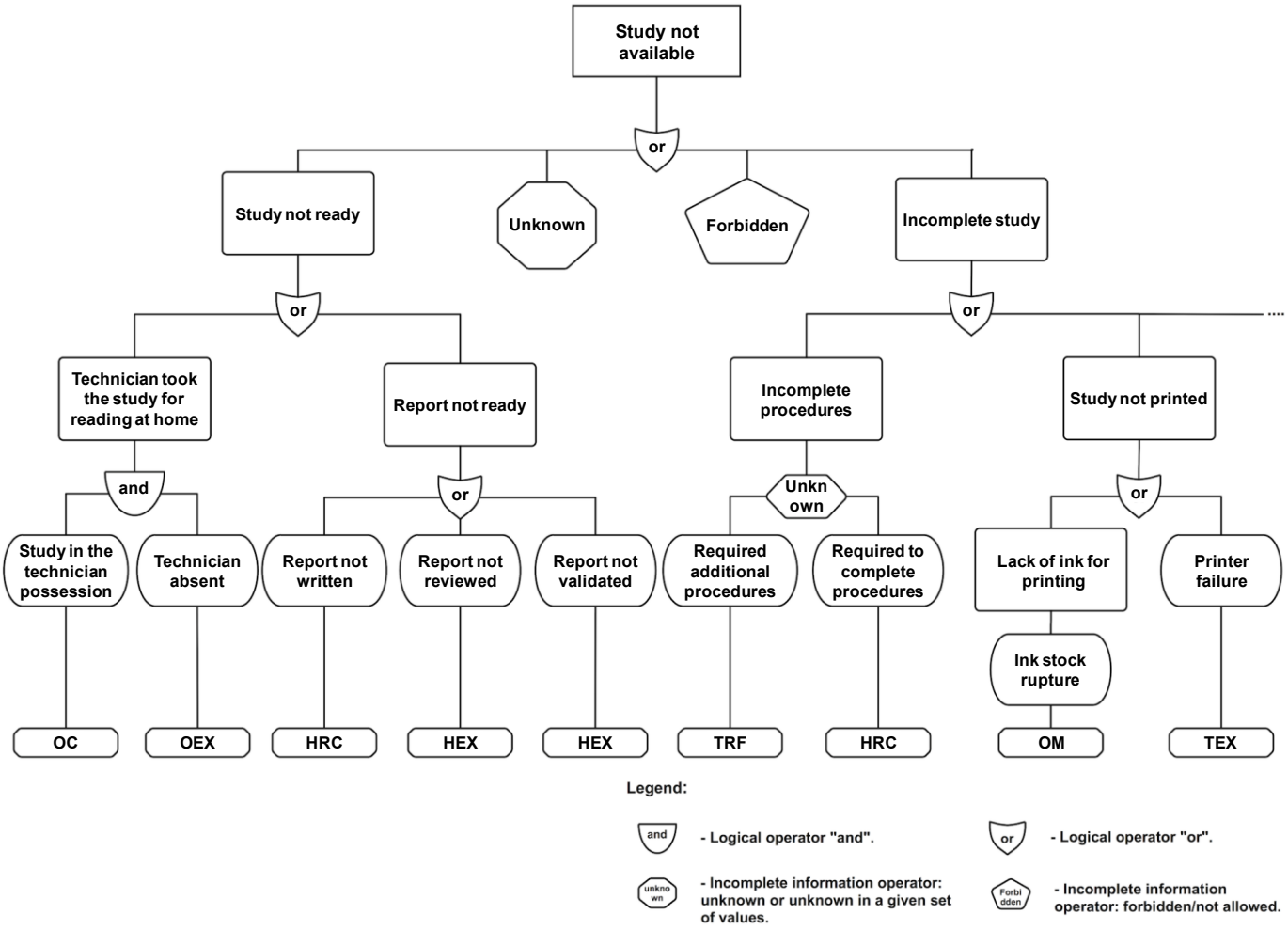

Fig. 2. Extended Causal Tree for the adverse event "*Study Not Available*"

ELP introduces another kind of negation, strong negation, represented by the classical negation sign ¬. In most situations, it is useful to represent ¬*p* as a literal, if it is possible to prove ¬*p*. In *ELP*, the expressions *p* and *not p*, being *p* a literal, are extended literals, while *p* or ¬*p* are simple literals. Intuitively, *not p* is true whenever there is no reason to believe *p*, whereas ¬*p* requires a proof of the negated literal [10].

Every program is associated with a set of *abducibles*, which may be seen as *hypotheses* that provide possible solutions or explanations of given queries, being given here in the form of *exceptions* to the extensions of the predicates that make the logical program or theory. The issue is providing expressive power for representing explicitly negative information, as well as to directly describe the closed world assumption for some predicates, also known as predicate circumscription [11].

Three types of answers to a given question are then possible, i.e., *true*, *false* and *unknown*. The representation of null values will be scoped by the *ELP*. It is possible to consider three types of null values: the former will allow for the representation of unknown values, not necessarily taken from a given set of values, the middle one will represent unknown values taken from a given set of possible values, and the latest will define values that are not allowed or forbidden. Taking the example of the adverse event "*study not available*" (Fig. 2) it might represent all the possible situations according to the following setting:

– It is known that the study was not available because it was in the technician's possession – *known value*;

– The professional that recorded the adverse event only informed that the study report was not ready. It is not possible to be constructive, concerning the action or truth- -value to consider. However, it is false that the action or decision could be different. This situation suggests that the lack of knowledge may be associated to a set of possible known values – *unknown value in a finite set of values* (in this case there are three possibilities, i.e., *report not written*, *report not reviewed* or *report not validated*);

- It is only known that the study was not available. In this case who reported the adverse event did not know which actions or decisions led to the event occurrence – *unknown value*;
- And finally, namely due to internal policies of the organization, it is not permitted to know the causes of a given event – *forbidden or not allowed values*.

Considering the extensions of the predicates that represent the information expressed in a generic causal tree when the *EECM* is applied, where the first predicate denotes the adverse event that was reported (*adverse_event (study not available)*), the second represents an action or decision that led to the adverse event occurrence and the third concerns the root cause that was the primary factor that contribute to the actions and decisions taken and, consequently, to the event occurrence:

- *adverse_event: X*
- *action_or_decision: Y*
- *root_cause: Z*

The knowledge representation in terms of the extension of predicate *action_or_decision*, concerning possible action or decision that leads to the adverse event in the situations presented above, may be depicted by the following programs.

Program 1. Extension of predicate *action_or_decision*, concerning a possible action or decision that leads to the adverse event "*study not available*", with a *known value*.

```
¬ action_or_decision(Y)←
                    not action_or_decision(Y),
                    not exception(action_or_decision(Y)).
action_or_decision("study in the technician possession").
```

Program 2. Extension of predicate *action_or_decision*, concerning a possible action or decision that leads to the adverse event "*study not available*", with an *unknown value in a finite set of values*.

```
¬ action_or_decision(Y)←
                    not action_or_decision(Y),
                    not exception(action_or_decision(Y)).
exception(action_or_decision("report not written")).
exception(action_or_decision("report not reviewed")).
exception(action_or_decision("report not validated")).
```

Program 3. Extension of predicate *action_or_decision*, concerning a possible action or decision that leads to the adverse event "*study not available*", with an *unknown value*, were ⊥ stand for a null value of an undefined type.

```
action_or_decision(⊥).
¬ action_or_decision(Y)←
                    not action_or_decision(Y),
                    not exception(action_or_decision(Y)).
exception(action_or_decision(Y))←
                    action_or_decision(⊥).
```

Program 4. Extension of predicate *action_or_decision*, concerning a possible action or decision that leads to the adverse event "*study not available*", with a *value forbidden* or *not allowed*.

```
action_or_decision(forbidden).
¬ action_or_decision(Y)←
                    not action_or_decision(Y),
                    not exception(action_or_decision(Y)).
exception(action_or_decision(Y))←
                    action_or_decision(forbidden)
null(forbidden).
```

Using *ELP*, as the logic programming language, it is now possible to set a procedure given in terms of the extension of a predicate called *demo: question, answer* → [0, 1]. Given a question *(Q)*, it returns a solution based on a set of assumptions, where question indicates a theorem to be proved and answer denotes a truth value (see Program 5; *True* (*1*), *False* (*0*), being *Unknown (U)* in the range of the truth values in the interval]0, 1[).

Program 5. Extension of meta-predicate demo.

```
demo(Q,T) ← Q
demo(Q,F) ← ¬Q
demo(Q,U) ← not Q ∧ not ¬Q
```

D. Quality of Information

The *Quality of Information* (*QoI*) factor with respect to the extension of a generic predicate *p* may be analysed in different contexts and measured in the interval [0, 1]. When the information is known; when the information is unknown; when the information is unknown but can be taken from a set of values. If the information is *known* the QoI_p for the extension of predicate *p* is 1. For situations where the value is *unknown* the QoI_p is given by:

$$QoI_p = \lim_{N \to \infty} \frac{1}{N} = 0 \, (N >> 0) \tag{1}$$

Finally, if the information is *unknown* but can be derived from a set of values, the QoI_p is set in terms of *1/Card*, where *Card* denotes the cardinality of the *abducibles* set for *p*.

The next element of the model to be considered is the relative importance that a predicate assigns to each of its attributes under observation, i.e., w_{ij} stands for the relevance of attribute *j* for predicate *i*. Assuming that the weights of all predicates are normalized, it is now possible to define a predicate's *scoring function* ($V_i(x)$), i.e., for a value $x = (x_1, ..., x_n)$ in the multi-dimensional space defined by the attributes domains, which is given in the form:

$$V_i(x) = \sum_{j=1}^{n} w_{ij} * V_{ij}(x_j) \tag{2}$$

It is viable to measure the *QoI* that occurs as a result of invoking a logic program to prove a theorem, by posting the

$V_i(x)$ values into a multi-dimensional space and projecting it onto a two dimensional one. Using this procedure, a circle with dashed n-slices can be defined denoting the *QoI* that is associated with each one of the predicate extensions that make the logic program.

As an example the *QoI* associated with the information about the *RCA* of the adverse event "*study not available*", for the first three cases present in the previous section, is given in the form:

- $V_{action\_or\_decision}$ *(former case)* = 1
- $V_{action\_or\_decision}$ *(middle term case)* = 0.33
- $V_{action\_or\_decision}$ *(latest case)* = 0

Thereby it is possible to measure the *QoI* associated to the question put in context, in terms of a logic program that endorses procedures of *action_or_decision*, which may be given in the form *Which are the actions or decisions that led to the adverse event occurrence?*. The shaded n-slices (here n is equal to 3 (three)) of the circle depicted in Figure 3 denote the *QoI*.

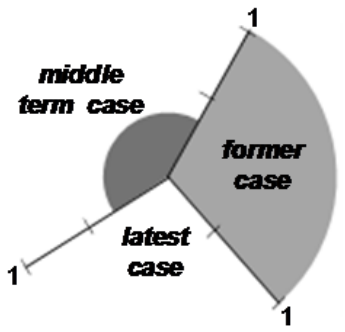

Fig 3. The embedded QoI with respect to the question Which are the actions or decisions that led to the adverse event occurrence?

III. Discussion

Based on the formal approach referred to above, an adverse event reporting and learning system was introduced. Indeed, to the professionals of conservation and maintenance of buildings and mostly to the organizations of the sector, this approach may bring some advantages. After the adverse events have been registered, similar to what happens in other reporting systems, the analysis process becomes easier, more expedite and reliable.

Undoubtedly, with the recourse to *ELP*, leading to an on the fly measurement of the *QoI* of the logic terms used in the process of judgement (in terms of a theorem to be proved), the human intervention in the analyse process is only necessary to approve the recommendations, causes and events that may need attention. It also caters for the credibility and the measurement of the efficacy of the implemented strategies and actions.

Although the causal classification of events is sometimes time-consuming and difficult, with the development of a generic causal tree for each possible event, the increase in time consuming is on the initial phase of the model enforcement.

The *QoI* allows the ordering of causes, identifying the ones that should be taken into account in the first place. In the generic tree it is necessary to consider all possible causes, rather than the most probable or usual ones. The information obtained is useful in identifying possible trends and areas requiring further investigation.

The conceptualized logic model offers the means for knowledge extraction, providing the identification of the most significant causes and suggestions of changes in the organization policies and maintenance procedures, subject to formal proof. Indeed, the creation of an inference system in support of the logical model enables the generation of reports with strategies for quality improvement on time, where a quality measure of the system is on one´s confidence on the results, in terms of the *QoI*.

IV. Conclusion

The main contribution of this work is to be understood in terms of the evaluation of the *QoI* in the RCA and the possibility to address the issue of incomplete information, through the use of an *Extension to Logic Programming* (*ELP*) in the construction of causal trees. *ELP* was used for knowledge representation and reasoning with defective information, catering for the modelling of the universe of discourse in terms of incomplete, inconsistent, forbidden and default data, information and knowledge.

A systematisation of the body of knowledge's evolution about *QoI* embedded in the *RCA* was accomplished. A way to solve the representation problem of defective information was presented, adequate for evaluating the *QoI* in such situations. It was also presented a computationally feasible formal tool to measure the value of *QoI*. With this approach to *RCA* and classification it was possible to identify the causes, actions and decisions that may lead to the adverse events and define the strategies to prevent them.

V. Future Work

In the future an *Adverse Event reporting and learning System applied to the Conservation and maintenance of Buildings* (*AESCB*) will be developed. The *AESCB* will comprise 3 (three) core modules, making it not only a system for adverse event registration, but also a learning system. The *Adverse Event Reporting Forms in Conservation and maintenance of Buildings* (*AERFCB*) module will provide a *Web* interface for adverse event registration.

The effort on this interface will be focused in its usability. The event registration will be made by professionals of the sector of conservation and maintenance of buildings and by those who use the buildings, through predefined forms adapted to each user profile.

The *Adverse Events Manager Reports in Conservation and maintenance of Buildings* (*AEMRCB*) module will be also *Web* based and aims to enable the analysis of the adverse events recorded by *AERFCB*, based on the *Extension of the Eindhoven Classification Model* (*EECM*). The system will provide an individual report for each adverse event recorded, which will include all its details and the extended causal tree obtained using the *EECM*.

The *AEMRCB* module will also provide charts with statistical information about the impact, place of occurrence, type of form and type of event recorded. Finally, the *Adverse Events Knowledge Manager in Conservation and maintenance of Buildings* (*AEKMCB*) module will use the data from the system database to create a *Knowledge Base* (*KB*), which although had been given in terms of *ELP*, will be rewritten to productions in the logic programming language *PROLOG* [12], based on the *EECM*.

From the *KB* other reports relevant to the improvement of the repair strategies may be generated, always with the assurance of data reliability and credibility, by taking into account its *QoI*.

ACKNOWLEDGMENTS

This work is funded by ERDF—European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT—Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028980.

REFERENCES

[1] H. Adeli, "Neural Networks in Civil Engineering: 1989–2000," *Computer--Aided Civil and Infrastructure Engineering*, vol. 16, pp. 126-142, 2001.

[2] P. Lu, S. Chen and Y. Zheng "Artificial Intelligence in Civil Engineering," *Mathematical Problems in Engineering*, vol. 2012. [Online]. Available: http://www.hindawi.com/journals/mpe/2012/145974/

[3] D. Dukić, M. Trivunić and A. Starčev-Ćurčin, "Computer-Aided Building Maintenance with "BASE-FM" Program," *Automation in Construction*, vol. 30, 57-69, 2013.

[4] I. Motawa and A. Almarshad, "A Knowledge-Based BIM System for Building Maintenance," *Automation in Construction*, vol. 29, 173-182, 2013.

[5] T. W. van der Schaaf, "Near Miss Reporting in the Chemical Process Industry: An Overview," *Microelectronics Reliability*, vol. 35, 1233-1243, 1995.

[6] T. W.van der Schaaf and M. Habraken, "PRISMA-Medical: A Brief Description," Eindhoven University of Technology, Faculty of Technology Management, Patient Safety Systems, 2005. [Online]. Available: http://www.who.int/patientsafety/taxonomy/PRISMA_Medical.pdf

[7] S. Rodrigues, P. Brandão, L. Nelas, J. Neves, and V. Alves, "A Logic Programming Approach to Medical Errors in Imaging," *International Journal of Medical Informatics*, vol. 80, 669-679, 2011.

[8] F. Sheridan, "A Survey of Techniques for Inference under Uncertainty," *Artificial Intelligent Review*, vol. 5, 89-119, 1991.

[9] M. L. Ginsberg, *Readings in Nonmonotonic Reasoning*. San Francisco: Morgan Kauffman Publishers Inc, 1987.

[10] J. Neves, "A logic interpreter to handle time and negation in logic data bases," in Proceedings of the 1984 annual conference of the ACM on the fifth generation challenge, R. L. Muller and J. J. Pottmyer, Eds. New York: Association for Computing Machinery, 1984, pp. 50-54.

[11] U. Hustadt, "Do we need the Closed-World Assumption in Knowledge Representation?," in Working Notes of the KI'94 Workshop – Reasoning about Structured Objects: Knowledge Representation meets Databases (KRDB'94), F. Baader, M. Buchheit, M. A. Jeusfeld and W. Nutt, Eds. Saarbrüken: German Research Center for Artificial Intelligence, 1994, pp. 24-26.

[12] I. Bratko, *PROLOG Programming for Artificial Intelligence*. Toronto: Pearson Education Canada, 2011.

Merging Deductive and Abductive Knowledge Bases: An Argumentation Context Approach

Juan Carlos Nieves and Helena Lindgren

***Abstract*—The consideration of heterogenous knowledge sources for supporting decision making is key to accomplish informed decisions, *e.g.*, about medical diagnosis. Consequently, merging different data from different knowledge bases is a key issue for providing support for decision-making. In this paper, we explore an argumentation context approach, which follows how medical professionals typically reason, in order to merge two basic kinds of reasoning approaches based on logic programs: deductive and abductive inferences. In this setting, we introduce two kinds of argumentation frameworks: deductive argumentation frameworks and abductive argumentation frameworks. For merging these argumentation frameworks, we follow an approach based on argumentation context systems. We illustrate the approach by considering two different declarative specifications of evidence-based medical knowledge into logic programs in order to support informed medical decisions.**

***Index Terms*—Knowledge representation, deductive knowledge bases, abductive knowledge bases.**

I. Introduction

THE knowledge used when reasoning about a medical diagnosis is ideally based on evidence-based medical knowledge generalizable over a large population. However, this knowledge is translated into diagnostic criteria based on consensus among researchers in order to become applicable to a single individual and of practical use in the encounter with a patient. These different types of sources of knowledge make use of different reasoning strategies, which are co-existing and observable in medical professionals' decision making (e.g., causal and diagnostic reasoning) [1]. We acknowledge this, and propose the notion of *default argumentation context system*, meaning that there is at least two supporting perspectives for each claim, where the supplementary part of an argument may be considered being a *meta*-argument, providing strength based on contextual information. An illustrating example is the following: consider the situation where there are diagnostic criteria for a disease, which a patient partly fulfills considering the available observations. However, since the available knowledge is incomplete, a verification is made using an evidence-based medical study where the diagnosis can be supported based on a population study conducted in the area where the patient is living.

Manuscript received on August 1, 2013; accepted for publication on September 30, 2013.

The authors are with the Department of Computing Science, Umeå University, SE-901 87, Umeå, Sweden (e-mail: {jcnieves,helena}@cs.umu.se).

Nowadays in distributed systems, the integration of multiple knowledge bases has been taking relevance. Indeed, one can find different approaches in the state of the art (e.g. the context-based argumentation framework outlined in [2] and the multi-context systems in [3]). Both approaches aim at utilizing, and bridging different frameworks of interpretation of available observations.

Against this background, we will explore the multi-context systems approach further in this paper with the goal to combine different kinds of argumentation frameworks to generate *informed medical decisions*. To this end, we define two kinds of argumentation frameworks: *deductive and abductive argumentation frameworks*. The interaction between deductive and abductive argumentation frameworks is managed by the so called *bridge rules*. An illustration of this interaction is depicted in Figure 1. Both the deductive and abductive argumentation frameworks are based on logic programs with *negation as failure* and the *well-founded semantics* (WFS) [4]. In particular, we consider WFS for building deductive and abductive arguments. We want to point out that we chose WFS because this semantics is polynomial time computable; moreover, there are logic programming engines which compute WFS, *e.g.*, DLV[1], SMODELS[2],XSB[3]. In order to integrate the deductive and abductive argumentation frameworks, we introduce the so called *default argumentation context systems*. We show that by considering particular argumentation semantics as *the grounded semantics*, one can infer *collective acceptable states* from a default argumentation context system in polynomial time (Proposition 3).

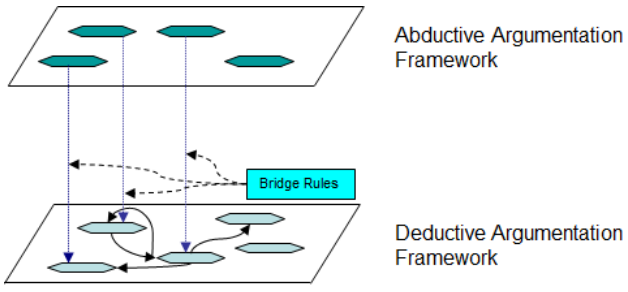

Fig. 1. Combining different knowledge bases by considering different kinds of argumentation frameworks.

[1]http://www.dbai.tuwien.ac.at/proj/dlv/
[2]http://www.tcs.hut.fi/Software/smodels/
[3]http://xsb.sourceforge.net/

As a running example, we will use an use-case when a patient shows symptoms that can be evaluated using more than one knowledge source and the diagnosis that is supported by more knowledge sources is preferred. The example involves two physicians, one is a novice and less experienced with the type of disease in focus for diagnosis, and one is an experienced physician. These two typically use different reasoning strategies [1], and consequently, they need different types of support.

The rest of the paper is divided as follows: In Section II, the syntaxis of the logic programs which we consider is introduced and the well-founded semantics is illustrated. In Section III, our main contribution is presented. Basically, we introduce all the components of the default argumentation context systems. In the last section, we outline our conclusions and future work.

II. Background

In this section, some basic concepts of logic programs are presented. In particular, the syntaxis of extended normal logic programs is presented. For capturing the semantics of these programs, the well-founded semantics ([4]) is considered, by lack of space we omit its formal definition. We assume that the reader is familiar with basic background on Logic Programming with *negation as failure*. A good introduction to Logic Programming with negation as failure can be found in [5]

A. Normal Logic Programs

The language of propositional logic has an alphabet consisting of:

(i) propositional symbols: $p_0, p_1, ...$;
(ii) connectives : $\vee, \wedge, \leftarrow, \neg, \ not, \top$;
(iii) auxiliary symbols : (,);

in which $\vee, \wedge, \leftarrow$ are 2-place connectives, $\neg$, not are 1-place connectives and $\top$ is a 0-place connective. The propositional symbols, $\top$, and the propositional symbols of the form $\neg p_i$ ($i \geq 0$) stand for the indecomposable propositions, which we call *atoms*, or *atomic propositions*. Atoms negated by $\neg$ will be called *extended atoms*. We will use the concept of atom without paying attention to whether it is an extended atom or not. The negation sign $\neg$ is regarded as the so called *strong negation* by the Answer Set Programming's literature and the negation not as the *negation as failure*. A literal is an atom, a (called positive literal), or the negation of an atom $not\ a$ (called negative literal).

A (propositional) extended normal clause, *C*, is denoted:

$$a \leftarrow b_1 \wedge \cdots \wedge b_j \wedge not\ b_{j+1} \wedge \cdots \wedge not\ b_{j+n} \qquad (1)$$

where $j+n \geq 0$, a is an atom and each b_i ($1 \leq i \leq j+n$) is an atom. When $j+n = 0$ the clause is an abbreviation of $a \leftarrow \top$ such that $\top$ is the propositional symbol that always evaluates to true. In a slight abuse of notation, we sometimes write the clause (1) as $a \leftarrow \mathcal{B}^+ \wedge not\ \mathcal{B}^-$, where $\mathcal{B}^+ := \{b_1, \ldots, b_j\}$ and $\mathcal{B}^- := \{b_{j+1}, \ldots, b_{j+n}\}$. An extended normal program P is a finite set of extended normal clauses. When $n = 0$, the clause is called *extended definite clause*. *An extended definite logic program* is a finite set of extended definite clauses. By $\mathcal{L}_P$, we denote the set of atoms in the signature of P. Let $Prog_{\mathcal{L}}$ be the set of all normal programs with atoms from $\mathcal{L}$.

We will manage the strong negation ($\neg$) in our logic programs as it is done in Answer Set Programming (ASP) [5]. Basically, each atom of the form $\neg a$ is replaced by a new atom symbol a' which does not appear in the language of the program. In order not to allow inconsistent models from logic programs, a normal clause of the form $f \leftarrow a \wedge a' \wedge\ not\ f$ such that $f \notin \mathcal{L}_P$ is added.

Example 1: We illustrate a normal logic program with an example from the dementia domain (simplified due to space reasons). A summary of the clinical guidelines, which are used in the dementia example given here can be found in [6]. We use the following abbreviations:

$$\begin{aligned}
AD &= \textit{Alzheimer's disease}\\
DLB &= \textit{Lewy body type of dementia}\\
VaD &= \textit{Vascular dementia}\\
epiMem &= \textit{Episodic memory dysfunction}\\
fluctCog &= \textit{Fluctuating cognition}\\
fn &= \textit{Focal neurological signs}\\
prog &= \textit{Progressive course}\\
radVasc &= \textit{Radiology exam shows vascular signs}\\
slow &= \textit{Slow, gradual onset}\\
extraPyr &= \textit{Extrapyramidal symptoms}\\
visHall &= \textit{Visual hallucinations}
\end{aligned}$$

By considering the previous abbreviations as propositional atoms, let P be a normal logic program formed by the following set of normal clauses.

1) $VaD \leftarrow fn \wedge\ not\ AD \wedge\ not\ DLB$
2) $VaD \leftarrow radVasc \wedge\ not\ AD \wedge\ not\ DLB$
3) $AD \leftarrow slow \wedge prog \wedge epiMem \wedge\ not\ VaD \wedge\ not\ DLB$
4) $DLB \leftarrow extraPyr \wedge visHall \wedge not\ fn$
5) $DLB \leftarrow fluctCog \wedge visHall \wedge not\ fn$
6) $DLB \leftarrow fluctCog \wedge extraPyr \wedge not\ fn$
7) $VaD \leftarrow fn \wedge radVasc$

These normal clauses will be considered for building arguments in the following sections.

B. Logic semantics

In this section, we present basic intuitions of 3-valued logic programming semantics. To this end, we present a basic definition of a 3-valued logic programming semantics.

Definition 1 (SEM [7]): For a normal logic program P, we define HEAD(P) $:= \{a|\ a \leftarrow \mathcal{B}^+ \wedge\ not\ \mathcal{B}^- \in P\}$ — the set of all head-atoms of P. We define SEM(P) $= \langle P^{true}, P^{false} \rangle$, where $P^{true} := \{p|\ p \leftarrow \top \in P\}$ and $P^{false} := \{p|\ p \in \mathcal{L}_P \backslash \text{HEAD}(P)\}$. SEM($P$) is called a model of P.

Basically, the 3-valued model SEM(P) of a given logic program P identifies three different classes of atoms:

1) Atoms which are considered *true w.r.t.* SEM(P), *i.e.* atoms from $\mathcal{L}_P$ which belong to P^{true};
2) Atoms which are considered *false w.r.t.* SEM(P), *i.e.* atoms from $\mathcal{L}_P$ which belong to P^{false};
3) Atoms which are considered *undefined w.r.t.* SEM(P), *i.e.* atoms from $\mathcal{L}_P$ which do not belong to $P^{true} \cup P^{false}$.

In the logic programming literature, there are different 3-valued logic programming semantics which have been intensively studied [8], [9]. Among these logic programming semantics, the Well-Founded Semantics (WFS) [4] satisfies well-expected properties about non-monotonic reasoning [9]. Indeed, WFS is *a well-behaved semantics* [9] and is regarded as an approximation of the Sable Model Semantics [10] which is the core the *Answer Set Programming* paradigm. An important computational property of WFS is that it is polynomial time computable. Nowadays, there are several solvers which can compute WFS in an efficient way: DLV[4], SMODELS[5],XSB[6]. In this paper, we will use the inference of WFS. By lack of space, we omit the formal definition of WFS; however, the reader can find the WFS's definition in [4] and use one of the mentioned solvers for computing WFS. Given a normal logic program, $WPS(P)$ will denote the well-founded model of P. Like SEM(P), $WFS(P)$ is basically a 3-valued model. In order to illustrate $WFS(P)$, let us consider the following example:

Example 2: Let P be the following normal logic program:

$$b \leftarrow not\ a. \qquad c \leftarrow not\ b. \qquad c \leftarrow a.$$

By using a WFS solver, we can see that $WFS(P) = \langle\{b\}, \{a, c\}\rangle$. This means that the atom b is true according with $WFS(P)$; on the other hand, the atoms a and b are false according with $WFS(P)$.

In order to simplify to the presentation of some definitions (in the next sections), we introduce some notation. Let P be a normal logic program and $WFS(P) = \langle T, F\rangle$ be the well-founded model of P. Hence

- $P \models_{WFS^T} a$ if and only if $a \in T$;
- $P \models_{WFS^F} a$ if and only if $a \in F$

For instance, by considering the normal program P and $WFS(P)$ from Example 2, we can see that $P \models_{WFS^T} b$, $P \models_{WFS^F} a$ and $P \models_{WFS^F} c$.

III. Default Argumentation Context Systems

In this section, the idea of *Default Argumentation Context Systems* will be defined. To this end, two kinds argumentation frameworks will be defined: *deductive argumentation frameworks* and *abductive argumentation frameworks*. The idea is that given a set of observations a deductive argumentation framework, built from a deductive knowledge base, will infer conclusions. On the other hand, given the conclusions of the deductive argumentation framework, an abductive argumentation framework, built from an abductive knowledge base, will support the conclusions of the given deductive argumentation framework. In order to merge the two kinds of argumentation framework, we will follow the ideal of *Argumentation Context Systems* which were introduced in [11]. We will start defining deductive argumentation frameworks.

A. Deductive Argumentation Frameworks

The structure of deductive argumentation frameworks (DAFs) will follow the well-known structure of *argumentation frameworks* which were introduced by Dung [12]. Hence, a DAF basically is a set of deductive arguments and a set of attacks between them. Therefore, let us start defining deductive arguments.

Definition 2 (Deductive argument): Let P be an extended normal logic program and $O \subseteq \mathcal{L}_P$ such that O is called observations. $A_D = \langle S, O', c\rangle$ is a deductive argument if the following conditions holds:

1) $S \cup O' \models_{WFS^T} c$,
2) $S \subseteq P$ such that S is a minimal set among the subsets of P satisfying 1,
3) $O' \subseteq O$ such that O' is a minimal set among the subsets of O satisfying 1.
4) $WFS(S \cup O') = \langle T, F\rangle$ such that $\nexists a \in \mathcal{L}_P$ and $\{a, \neg a\} \subseteq T$.

$\mathcal{A}^D(P, O)$ denotes the set of deductive arguments built from P and O.

As we can observe in Definition 2, a deductive argument basically is a tuple of the form $\langle S, O', c\rangle$. The first condition of the definition suggests that $S \cup O'$ is the support of the argument and c is the claim of the argument. From conditions 2 and 3, we can guarantee that the support of the argument is the minimum information which can infer c by considering WFS. The last condition guarantees that this support is consistent.

Let us consider the following scenario in order to illustrate the definition of deductive arguments: An older person comes to the emergency room, after having fallen in her home. This has happened several times lately, and she cannot understand why and is getting very worried. In addition, she has difficulties in performing activities in her daily life, and it turns out that she has a state of dementia. However, this does not explain why she is falling, so the physician finds a reason to investigate further. The physical and cognitive examinations result in finding extrapryamidal symptoms (parkinsonism but without having a Parkinson's disease diagnosis), focal neurological symptoms, in addition to that her cognitive functions tend to fluctuate during the day. At this point the experienced physician is able to generate a set of hypothetical diagnoses containing the correct one, based on the information. The less experienced typically has difficulties using this approach to generate hypotheses [1], but may utilize

[4]http://www.dbai.tuwien.ac.at/proj/dlv/
[5]http://www.tcs.hut.fi/Software/smodels/
[6]http://xsb.sourceforge.net/

a decision support application, which behaves in the following way.

Example 3: Let P be the normal program introduced in Example 1 and $O = \{fn,\ extraPyr,\ fluctCog\}$. Some deductive arguments which one can build from P and O are:

$Arg_D^1 = \langle \{\}, \{fn\}, fn \rangle$
$Arg_D^2 = \langle \{DLB \leftarrow fluctCog \wedge extraPyr \wedge not\ fn\}, \{fluctCog, extraPyr\}, DLB \rangle$
$Arg_D^3 = \langle \{VaD \leftarrow fn \wedge\ not\ AD \wedge\ not\ DLB\}, \{fn\}, VaD \rangle$

From these arguments, we can believe that the given patient could be diagnosed with *Lewy body type dementia* (DLB) or *Vascular dementia* (VaD). However, these arguments are not final decisions. In order to have candidate decisions, we require to consider their disagreements and to select potential *acceptable* deductive arguments.

Once we have defined the structure of a deductive argument, the attack relation between these arguments is defined as follows:

Definition 3 (Attack relation between deductive arguments): Let $A = \langle S_A, O_A, c_A \rangle$, $B = \langle S_B, O_B, c_B \rangle$ be two deductive arguments, $WFS(S_A \cup O_A) = \langle T_A, F_A \rangle$ and $WFS(S_B \cup O_B) = \langle T_B, F_B \rangle$. We say that A attacks B if one of the following conditions holds:

- $a \in T_A$ and $\neg a \in T_B$.
- $a \in T_A$ and $a \in F_B$.

$At^D(S)$ denotes the set of attack relations between the deductive arguments which belong to a set of deductive arguments S.

Observe that the first condition of the definition is capturing the standard idea of *rebut* and the second condition is capturing the standard idea of *undercut*. Rebut and undercut are two well accepted ideas of disagreement between arguments in the argumentation theory [13].

Example 4: Let us consider the three deductive arguments which were introduced in Example 3. One can see the following relations of attack:

Arg_D^1 attacks Arg_D^2 $\qquad$ Arg_D^2 attacks Arg_D^3

Now we are in position for introducing the definition of a deductive argumentation framework.

Definition 4 (Deductive Argumentation Framework): Let P be an extended normal logic program and $O \subseteq \mathcal{L}_P$. A deductive argumentation framework is a tuple of the form $\langle \mathcal{A}^D(P,O), At^D(\mathcal{A}^D(P,O)) \rangle$.

As we can see, a deductive argumentation framework basically follows the structure of Dung's argumentation frameworks. The main difference is that a deductive argument has a structure which is based on logic programs with negation as failure and the inference of WFS. On the other hand, the attacks relation between deductive arguments is based on the inference of WFS.

It is quite easy to see that given the structure of a deductive argumentation framework, one can use an *extension-based argumentation semantics* [12] for selecting sets of acceptable deductive arguments from a deductive argumentation framework.

Example 5: Let AR^D be the deductive arguments introduced in Example 3. Hence $DAF = \langle AR^D, At^D(AR^D) \rangle$ is a deductive argumentation framework. $At^D(AR^D)$ is composed by the two attack relations identified in Example 4. By considering the grounded semantics (introduced in [12]), we can see that $\{Arg_D^1, Arg_D^3\}$ is the grounded extension[7] of DAF. By observing Arg_D^3, a physician can believe that the given patient could has Vascular dementia (VaD). The question here is: *how can a physician validate this diagnosis?* We will give an answer to this question in the next sections.

An interesting property of WFS is that this logic programming semantics satisfies *relevance* [9]. This property takes importance from the argumentation point of view in order to show that the join of the pieces of knowledge which support each argument which belongs to an extension is *consistent*[8]. The only requirement of the given extension is that has to be conflict-free, *e.g.*, the extension does not contain two arguments which attach each other. Hence, we say that an extension-based argumentation semantics s satisfies conflict-freeness if each extension which is inferred by s is conflict-free.

Proposition 1: Let P be an extended normal logic program, $O \subseteq \mathcal{L}_P$, $DAF = \langle \mathcal{A}^D(P,O),\ At^D(\mathcal{A}^D(P,O)) \rangle$ be a deductive argumentation framework and s be an extension-based argumentation semantics which satisfies conflict-freeness. If $E \in s(DAF)$, $P_E = \{S \cup O | \langle S, O, c \rangle \in E\}$ and $C_E = \{c | \langle S, O, c \rangle \in E\}$ then the following conditions hold:

1) $P_E \models_{WFS^T} c$ such that $c \in C_E$
2) There is not $c \in \mathcal{L}_P$ such that $P_E \models_{WFS^T} c$ and $P_E \models_{WFS^T} \neg c$

B. Abductive Argumentation Frameworks

The other kind of argumentation frameworks that we will consider are the so called *abductive argumentation frameworks*. The novice physician in our example may have had difficulties finding hypothetical diagnoses using the deductive reasoning approach. Instead, he considers the dementia diagnoses which he is familiar with, and identifies their effects which match his observations.

Like deductive argumentation frameworks, abductive argumentation frameworks are based on logic programs and the inference of WFS. However, the logic programs which are considered for building abductive argumentation frameworks will be a particular class of logic programs which are called *Abductive Logic Programs*.

Definition 5 (Abductive Program): Let P an extended logic program. An abductive logic program is a pair $\langle P, H \rangle$ where the following conditions hold:

[7] An extension is a set of arguments which is suggested by an extension-based argumentation semantics.

[8] We say that a logic program P is consistent if there is not a model of P which contains a and $\neg a$.

1) $H \subset \mathcal{L}_P$, H will be called hypothesis.
2) P is an extended normal logic program such $HEAD(P) \cap H = \emptyset$.

This definition follows the ideas of abductive programs introduced in [14]. Hence, by considering this definition an abductive argument is defined as follows:

Definition 6 (Abductive Argument): Let $P_{Ab} = \langle P, H\rangle$ be an abductive logic program and O a set of atoms. An abductive argument with respect to an atom $a \in O$ is $A^{Ab}(a) = \langle S, E, a\rangle$ such that the following conditions holds:

- $S \cup E \models_{WFS^T} a$
- $S \subseteq P$, $E \subseteq H$ and both S, E are minimal amount the subsets of P and E respectively.

$\mathcal{A}^{Ab}(P_{Ab}, A)$ denotes the set of abductive arguments built from P_{Ab} and A.

As the deductive argument (see Definition 2), an abductive argument basically is a tuple of the form $\langle S, E, a\rangle$ in which we can find a support of the argument ($S \cup E$) and a claim a. The definition of an abductive argument also requests that the support of the argument has to be minimal. Unlike to deductive arguments which take observations as part of their support, abductive arguments take hypothesis as part of their support. Indeed, we can onserve that an abductive argument explains an observation $a \in O$.

In the following example, we are going to illustrate abductive arguments.

Example 6: By considering the propositional atoms introduced in Example 1, let $P_{Ab} = \langle P', H\rangle$ be an abductive program such that $H = \{DLB, VaD, AD\}$, and P' the following set of normal clauses:

$extraPyr \leftarrow DLB.$
$fluctCog \leftarrow DLB.$
$visHall \leftarrow DLB.$
$fn \leftarrow VaD.$
$radVasc \leftarrow VaD.$
$epiMem \leftarrow AD.$

These normal clauses were defined by re-interpreting the clinical guidelines in order to explore what information they give on causality, *i.e.*, what can we expect to observe in an individual with a certain disease.

Let us consider the findings obtained in Example 3 $O = \{fn,\ extraPyr,\ fluctCog\}$ such that we want to find an explanation for each element of O by considering P_{Ab}. Hence, some abductive arguments from P_{Ab} and O are:

$Arg_1^{Ab} = \langle\{extraPyr \leftarrow DLB\}, \{DLB\}, extraPyr\rangle$
$Arg_2^{Ab} = \langle\{fluctCog \leftarrow DLB\}, \{DLB\}, fluctCog\rangle$
$Arg_3^{Ab} = \langle\{fn \leftarrow VaD\}, \{VaD\}, fn\rangle$

The first argument argues (Arg_1^{Ab}) that *extrapyramidal symptoms* ($extraPyr$) can be *explained by Lewy body dementia* (DLB). The last two argument have easy readings. Here the experienced physician is able to verify his hypotheses obtained using the deductive approach. The less experienced physician will limit his reasoning to the diagnoses he is familiar with, thus may miss the less common alternative DLB and jump to the conclusion that VaD is the cause of the symptoms [1] which is an insufficiently informed decision.

Due to the aim of abductive arguments in our particular application, which is to support the claims of deductive arguments, we will not define a *particular* attack relation (disagreements) between abductive arguments. In this setting, abductive argumentation frameworks will be defined as follows:

Definition 7 (Abductive Argumentation Frameworks): Let $P_{Ab} = \langle P, H\rangle$ be an abductive logic program and O be a set of atoms. An abductive argumentation framework *w.r.t.* P_{Ab} and O is defined as follows: $\langle \mathcal{A}^{Ab}(P_{Ab}, O), At\rangle$ such that $At \subseteq \mathcal{A}^{Ab}(P_{Ab}, O) \times \mathcal{A}^{Ab}(P_{Ab}, O)$.

Like deductive argumentation frameworks, abductive argumentation frameworks follow the structure of Dung's argumentation frameworks. Hence one can use extension-based argumentation semantics for selecting acceptable abductive arguments from an abductive argumentation framework. Indeed, one can formulate a version of Proposition 1 for abductive argumentation frameworks.

Proposition 2: Let $P_{Ab} = \langle P, H\rangle$ be an abductive logic program, O be a set of atoms, $AAF\langle\mathcal{A}^{Ab}(P_{Ab}, O), At\rangle$ be an abductive argumentation framework and s be an extension-based argumentation semantics which satisfies conflict-freeness. If $E \in s(AAF)$, $P_E = \{S \cup H | \langle S, H, c\rangle \in E\}$ and $C_E = \{c | \langle S, H, c\rangle \in E\}$ then the following conditions hold:

1) $P_E \models_{WFS^T} c$ such that $c \in C_E$
2) There is not $c \in \mathcal{L}_P$ such that $P_E \models_{WFS^T} c$ and $P_E \models_{WFS^T} \neg c$

C. Argumentation Context Systems

So far we have instantiated a deductive knowledge based into a deductive argumentation framework and an abductive knowledge base into an abductive argumentation framework. In order to support informed decision making (*e.g.*, medical diagnosis in our running example with the two differently experienced physicians) these two argumentation frameworks need to be combined, or merged.

To this end, we will follow the approach of *Argumentation Context Systems* [11]; in particular, we will define the idea of *default argumentation context systems*. For this purpose, we introduce a definition of *context expressions* and *contexts* which are key features of argumentation context systems.

From here whenever we use only the word argument, it means that this argument can be either a deductive argument or an abductive argument.

Definition 8 ([11]): A context expression for a set of arguments AR and a set of values V has one of the following forms ($a, b \in AR$; $v, v' \in V$):

$arg(a)/\overline{arg(a)}$	a is a valid/invalid argument
$att(a,b)/\overline{att(a,b)}$	(a,b) is valid/invalid attack
$a > b$	a is strictly preferred to b
$val(a,v)$	the value of a is v
$v > v'$	value v is strictly better than v'
$mode(r)$	the reasoning mode $r \in \{skep, cred\}$
$sem(s)$	s is a chosen argumentation semantic [12]

A context C is a set of context expressions.

One can see that a context defines different aspects *w.r.t.* a set of arguments. For instance:

a) preferences between arguments;
b) validity/invalidity of specific arguments (and attacks);
c) a reasoning mode (either skeptical or credulous); and
d) an extension-based argumentation semantics (grounded, stable, preferred, *etc.*) [12].

Hence, by considering a given context one can modify a (deductive/abductive) argumentation framework as follows:

Definition 9: Let $AF = \langle AR, att\rangle$ be a (deductive/abductive) argumentation framework, V be a set of values and C be a context for AR and V. The $C^{\top}$-modification of AR is the (deductive/abductive) argumentation framework $AF^C = \langle AR^C, att^C\rangle$, where

- $AR^C = AR \cup \{\langle\{\},\{\},\top\rangle\}$
- att^C is the smallest relation satisfying the following conditions:
  1) if $att(a,b) \in C$, then $(a,b) \in att^C$,
  2) if $(a,b) \in att$, $\overline{att(a,b)} \notin C$, and $b \not>_C a$, then $(a,b) \in att^C$,
  3) if $\overline{att(a,b)} \in C$ or $(arg(b) \in C \wedge (a,b) att^C)$ then $(\langle\{\},\{\},\top\rangle, a) \in att^C$.

Let observe that the new argument $\langle\{\},\{\},\top\rangle$, which is non-attackable, basically is defeating invalid arguments as well as attackers of valid arguments.

Following with *argumentation context systems*, we will consider the $C^{\top}$-modification for defining the sets of acceptable arguments.

Definition 10: Let $AF = \langle AR, att\rangle$ be a (deductive/abductive) argumentation framework, V be a set of values and C be a context for AR and V such that $sem(s) \in C$, $mode(m) \in C$. A set of arguments $S \subseteq AR$ is an acceptable $C^{\top}$-extension for AF, if either:

- $m = cred$ and $S \cup \{\langle\{\},\{\},\top\rangle\}$ is a s-extension of AF^C, or
- $m = skep$ and $S \cup \{\langle\{\},\{\},\top\rangle\}$ is the intersection of all s-extensions of AF^C.

where s-extension is an extension (a set of arguments) according to the extension-based argumentation semantics s.

In order to define *default argumentation context systems*, let us introduce the so called *bridge rules*. To this end, let us define the following notation: Let AR be a set of arguments and Σ be a set of atomic symbols of the same cardinality of AR. $\bullet$ is a bijective function from AR onto Σ. We shall denote the image of $a \in AR$ under $\bullet$ as $a^{\bullet}$. A straightforward generalization of $\bullet$ over AR is: $AR^{\bullet} = \{a^{\bullet} | a \in AR\}$.

Given a set of argument AR, a bridge rule is a normal clause of the form:

$$s \leftarrow a_1^{\bullet} \wedge \cdots \wedge a_n^{\bullet} \wedge\ not\ a_{n+1}^{\bullet} \wedge \cdots \wedge\ not\ a_m^{\bullet} \qquad (2)$$

where s is a context expression and $a_i \in AR (1 \leq i \leq m)$.

By considering bridge rules, we define a mediator as follows:

Definition 11: Let AF_1 and AF_2 be arbitrary (deductive/abductive) argumentation frameworks. A mediator for AF_1 based on AF_2 is a tuple of the form: $Med = \langle E, R_1\rangle$, where E is a set of context expressions for AF_1 and R_1 is a set of bridge rule of the form (2) such that s is a context expression for AF_1 and the arguments are from AF_2.

Now that we have defined mediators between argumentation frameworks, a module M is a tuple of the form $\langle AF, M\rangle$ such that AF is a deductive/abductive argumentation framework and M is a mediator for AF based on a given deductive/abductive argumentation framework.

A default argumentation context system is a pair of modules which is defined as follows:

Definition 12: A default argumentation context system (DACS) is a tuple of the form: $DAF = \langle M_1, M_2\rangle$, where $M_1 = \langle AF_1, Med_1\rangle$, $M_2 = \langle AF_2, Med_2\rangle$, AF_1 is a deductive argumentation framework, AF_2 is an abductive argumentation framework, Med_1 is based on AF_2 and Med_2 is based on AF_1.

Given the implicity dependency between M_1 and M_2, one can see that given sets of acceptable arguments for M_1, the mediators define the consistent *acceptable context* for M_2 and viceversa.

In order to define the semantics of default argumentation context systems, the acceptable states of a given default argumentation context system will be defined as follows.

Definition 13: Let $DACS = \langle M_1, M_2\rangle$ be a default argumentation context System.

- A state of $DACS$ is a function S that assign each $M_i = \langle AF_i, Med_i\rangle$ a pair $S(M_i) = \langle Acc_i, C_i\rangle$ of a subset Acc_i of arguments of AF_i and a set C_i of context expressions for AF_i, $i \in \{1,2\}$.
- A state S is stable if (i) each Acc_i is an acceptable $C^{\top}$-extension of AF_i and C_i is an acceptable context for M_i, $i \in \{1,2\}$.

In order to define *a default argumentation context framework for our running example*, we are going to introduce some notations: Let P be a normal logic program, $P_{Ab} = \langle P', H\rangle$ be an abductive program and $O \subseteq \mathcal{L}_P$. If $a^d \in \mathcal{A}^D(P,O)$, then:

$h(a^d) := a^{Ab}$ if $\quad a^{Ab} \in \mathcal{A}^{Ab}(P',O)$, $a^{Ab} = \langle S, H, c'\rangle, c \in H$ and $c' \in O'$

Example 7: Let $DAF = \langle AR^D, At^D(AR^D)\rangle$ be the deductive argumentation framework introduced in Example 5 and AR^{Ab} be the set of abductive arguments introduced in

Example 6. One can see that $AAF = \langle AR^{Ab}, \{\}\rangle$ is an abductive argumentation framework.

Let R be the set of bridge rules $\{arg(a) \leftarrow b | a \in AR^{D}, b \in AR^{Ab}$ and $h(a) = b\}$. We can see that $R = \{arg(Arg_D^2) \leftarrow Arg_1^{Ab^\bullet}, arg(Arg_D^2) \leftarrow Arg_2^{Ab^\bullet}, arg(Arg_D^3) \leftarrow Arg_3^{Ab^\bullet}\}$. An intuitive reading of the first bridge rule suggests that the abductive argument Arg_1^{Ab} supports the conclusion of the deductive argument Arg_D^2. The same reading can be done with the rest of bridge rules.

Let $Med_1 = \langle\{sem(grounded), mode(skep)\}, R\rangle$ and $Med_2 = \langle\{sem(grounded), mode(skep)\}, \{\}\rangle$ be two mediators. Hence, a default argumentation context system $DACS_{running}$ of our running example can be: $DACS_{running} = \langle\langle DAF, Med_1\rangle, \langle AAF, Med_2\rangle\rangle$. The unique stable state of $DACS_{running}$ suggests that $Acc_1 = \{Arg_D^3\}$ and $Acc_2 = AR^{Ab}$. An interpretation of this stable state suggests that according with the default argumentation context system, the diagnosis of *Vascular dementia* (Vad) is supported by the deductive argumentation framework and the abductive argumentation framework. Let us remember that Vad was the conclusion in Example 5. However, this conclusion is now validated by Arg_3^{Ab} from AAF.

It is known that the computational cost of extension-based argumentation semantics is hight. Indeed the computational complexity of the decision problems of the extension-based argumentation semantics range from NP-complete to $\Pi_2^{(p)}$-complete [15]. One of the efficient extension-based argumentation semantics which exists is the grounded semantics. Hence, the use of the grounded semantics in each module of a default argumentation context system (DACS) implies to compute the stable state of DACS in a efficient way.

Proposition 3: Let $DACS = \langle M_1, M_2\rangle$ be a default argumentation context system. If M_1 and M_2 use the grounded semantics, then the acceptable states of $DACS$ are computable in polynomial time.

IV. Conclusions and future work

We have explored argumentation context systems in order to support informed decision making with hematogenous knowledge bases. To this end, we have introduced two kinds of argumentation frameworks: *deductive* argumentation frameworks and *abductive* argumentation frameworks. These argumentation frameworks are based on logic programs with negation as failure and WFS. For merging these argumentation frameworks, Default Argumentation Context Systems have been introduced and exemplified by a medical diagnostic example.

An interesting property of WFS is that this logic programming semantics satisfies *relevance* [9]. This property takes importance from the argumentation point of view in order to show that the join of the pieces of knowledge which support each argument which belongs to an extension is *consistent*. Hence, we showed that the knowledge which belongs to a set of either deductive or abductive arguments is consistent if this set of arguments belongs to an extension of an argumentation semantics which satisfies *conflict-free* (Proposition 1 and Proposition 2).

We also showed that by considering particular argumentation semantics as *the grounded semantics*, one can infer *collective acceptable states* from a default argumentation context system in polynomial time (Proposition 3).

We are particularly interested in supporting medical diagnosis in the demential domain. Hence, we argue that our approach *mimics* the reasoning process of an expert physician. In this context, we can say that the novice physician is able to use only fragments of the abductive knowledge base, which limits the quality of his assessments. By contrast, the expert physician is able to use both the deductive and abductive knowledge bases, which contributes to a holistic perspective on a patient case. Since the Default Argumentation Context Systems mimic the expert's reasoning, they may guide the novice physician into a reasoning process which generates more informed decisions at the same time as the physician develops his skills in diagnostic reasoning.

The quality of the informed decisions supported by the Default Argumentation Context Systems could be improved even further by integrating possibilistic values attached to hypotheses and preferences among knowledge sources. This will be our main focus for future work. Indeed, we have argued that the declarative specifications of medical guidelines require rich specification languages which could capture uncertain and incomplete information [16], [17].

Acknowledgment

This research is partly funded by VINNOVA (Sweden's innovation agency) and the Swedish Brain Power.

References

[1] V. L. Patel, D. R. Kaufman, and J. F. Arocha, "Emerging paradigms of cognition in medical decision-making," *Journal of Biomedical Informatics*, vol. 35, no. 1, pp. 52–75, 2002.

[2] H. Lindgren and P. Eklund, "Differential diagnosis of dementia in an argumentation framework," *Journal of Intelligent and Fuzzy Systems*, vol. 17, no. 4, pp. 387–394, 2006.

[3] G. Brewka, T. Eiter, and M. Fink, "Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources," in *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, ser. Lecture Notes in Computer Science, vol. 6565. Springer, 2011, pp. 233–258.

[4] A. V. Gelder, K. A. Ross, and J. S. Schlipf, "The well-founded semantics for general logic programs," *Journal of the ACM*, vol. 38, no. 3, pp. 620–650, 1991.

[5] C. Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge: Cambridge University Press, 2003.

[6] J. O'Brien, D. Ames, and A. Burns, Eds., *Dementia*. Arnold, 2000.

[7] J. Dix, M. Osorio, and C. Zepeda, "A general theory of confluent rewriting systems for logic programming and its applications," *Ann. Pure Appl. Logic*, vol. 108, no. 1-3, pp. 153–188, 2001.

[8] J. Dix, "A classification theory of semantics of normal logic programs: I. strong properties," *Fundam. Inform.*, vol. 22, no. 3, pp. 227–255, 1995.

[9] ——, "A classification theory of semantics of normal logic programs: II. weak properties." *Fundam. Inform.*, vol. 22, no. 3, pp. 257–288, 1995.

[10] M. Gelfond and V. Lifschitz, "The Stable Model Semantics for Logic Programming," in *5th Conference on Logic Programming*, R. Kowalski and K. Bowen, Eds. MIT Press, 1988, pp. 1070–1080.

[11] G. Brewka and T. Eiter, "Argumentation Context Systems: A Framework for Abstract Group Argumentation," in *Logic Programming and Nonmonotonic Reasoning (LPNMR 2009)*, ser. Lecture Notes in Computer Science, vol. 5753. Springer, 2009, pp. 44–57.

[12] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, no. 2, pp. 321–358, 1995.

[13] H. Prakken and G. A. W. Vreeswijk, "Logics for defeasible argumentation," in *Handbook of Philosophical Logic*, 2nd ed., D. Gabbay and F. Günthner, Eds. Dordrecht/Boston/London: Kluwer Academic Publishers, 2002, vol. 4, pp. 219–318.

[14] C. Sakama and K. Inoue, "An abductive framework for computing knowledge base updates," *TPLP*, vol. 3, no. 6, pp. 671–713, 2003.

[15] P. E. Dunne, "Computational properties of argument systems satisfying graph-theoretic constraints," *Artificial Intelligence*, vol. 171, no. 10-15, pp. 701–729, 2007.

[16] J. C. Nieves, M. Osorio, and U. Cortés, "Semantics for possibilistic disjunctive programs," *Theory and Practice of Logic Programming*, vol. 13, pp. 33–70, 0 2013. [Online]. Available: http://journals.cambridge.org/article_S1471068411000408

[17] J. C. Nieves and H. Lindgren, "Possibilistic Nested Logic Programs," in *Technical Communications of the 28th International Conference on Logic Programming (ICLP'12)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), A. Dovier and V. S. Costa, Eds., vol. 17. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 267–276. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2012/3628

Multiscale RBF Neural Network for Forecasting of Monthly Hake Catches off Southern Chile

Nibaldo Rodriguez, Lida Barba and Jose Miguel Rubio L.

***Abstract*—We present a forecasting strategy based on stationary wavelet transform combined with radial basis function (RBF) neural network to improve the accuracy of 3-month-ahead hake catches forecasting of the fisheries industry in the central southern Chile. The general idea of the proposed forecasting model is to decompose the raw data set into an annual cycle component and an inter-annual component by using 3-levels stationary wavelet decomposition. The components are independently predicted using an autoregressive RBF neural network model. The RBF neural network model is composed of linear and nonlinear weights, which are estimates using the separable nonlinear least squares method. Consequently, the proposed forecaster is the co-addition of two predicted components. We demonstrate the utility of the proposed model on hake catches data set for monthly periods from 1963 to 2008. Experimental results on hake catches data show that the autoregressive RBF neural network model is effective for 3-month-ahead forecasting.**

***Index Terms*—Neural network, forecasting, nonlinear least squares.**

I. Introduction

THE highly productive coastal upwelling zone off central / southern Chile (30–40^{o}S) sustains a strong fishery based on hake catches. The hake is highly important for economic development in the southern zone of Chile. One of the main goals of the fishery industry and the governments is to develop sustainable exploitation policies. However, fluctuations in the marine ecosystem complicate this task. To the best of our knowledge, few publications exist on multi-step-ahead forecasting models for fisheries resources. In recent years, linear regression models [1], [2] and artificial neuronal networks (ANN) [3]–[5] have been proposed for fisheries forecasting models. The disadvantage of models based on linear regressions is the supposition of stationarity and linearity of the time series of pelagic species catches. Although ANN allows modeling the non-linear behaviour of a time series, they also have some disadvantages such as the stagnancy of local minimum due to the steepest descent learning method and over-fitting problem. A multilayer perceptron neural network to improve the convergence speed and forecasting accuracy of anchovy and sardines catches off northern Chile was proposed by [3]–[5], which reported a coefficient of determination between 82% and 87%.

In this paper, the stationary wavelet transform (SWT) combined with autoregressive RBF neural network models are applied to improve accuracy forecasting of monthly hake catches of the southern zone of Chile. The advantage of the SWT in non-stationary time series analysis is their capacity to separate low frequency (LF) from high frequency (HF) components [6]–[12]. Whereas the LF component reveals long-term trends, the HF component describes short-term fluctuations in the time series. To separate these components is a key advantage in the proposed forecasting strategies, since the behaviour of each frequency component is more regular than the raw time series.

On the other hand, neural network techniques based on the nonlinear least square method have been accepted in several domains as a flexible modeling technique, suited for capturing nonlinear relationships between predictor variables and a response variable. Therefore, the proposed forecaster decomposes the raw data set into annual cycle component and interannual component, which are predicted independently using a RBF neural network model, whereas the final forecasting results are the sum of results obtained from a single model.

This paper is organized as follows. In the next section, we briefly describe the stationary wavelet transform and the forecasting model. The simulation results and performance evaluation are presented in Section 3 followed by conclusions in Section 4.

II. Neural forecasting model based on SWT

This section presents the proposed forecasting model for monthly hake catches off southern Chile. The proposed forecaster basically involves three stages. In the first stage, the original data set is decomposed into 3-level stationary wavelet decomposition to separate the annual cycle component and the interannual component. In the second stage, the annual cycle component and the interannual component are forecasted independently using a RBF neural network model with the separable nonlinear least squares (SNLS) algorithm. In the third stage, the future values are predicted by the co-addition of two predicted components.

Manuscript received on August 2, 2013; accepted for publication on September 30, 2013.

Nibaldo Rodriguez is with the School of Computer Engineering at the Pontificia Universidad Católica de Valparaíso, Av. Brasil 2241, Chile (e-mail: nibaldo.rodriguez@ucv.cl).

Lida Barba is with the School of Computer Engineering at the Universidad Nacional de Chimborazo, Av. Antonio Jose de Sucre, Km 1.5, Ecuador.

Jose Miguel Rubio L. is with the School of Computer Engineering at the Pontificia Universidad Católica de Valparaíso, Av. Brasil 2241, Chile.

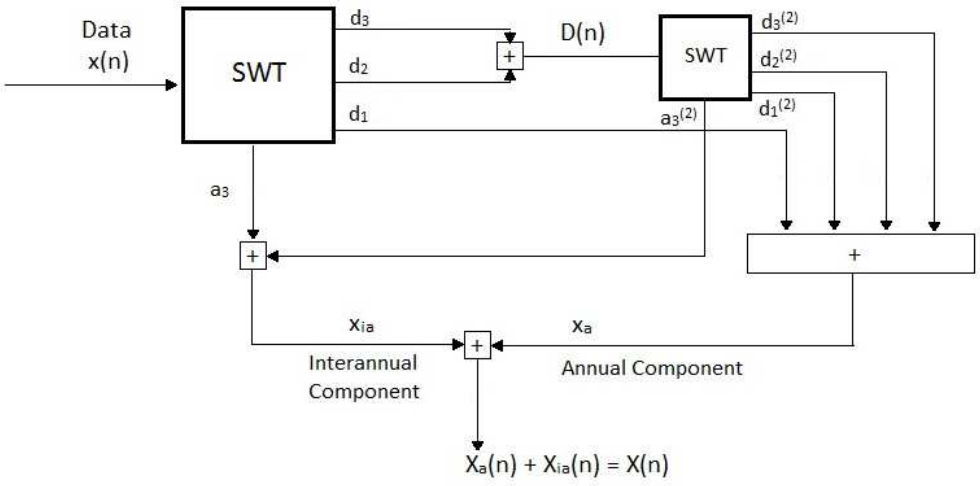

Fig. 1. Extraction process of components

A. Stationary wavelet decomposition

In time series analysis, discrete wavelet transform (DWT) often suffers from a lack of translation invariance. This problem can be tackled by means of the stationary wavelet transform (SWT). The SWT is similar to the DWT in that the high-pass and low-pass filters are applied to the input signal at each level, but the output signal is never decimated. Instead, the filters are up-sampled at each level.

A signal can be represented at multiple resolutions by decomposing the signal on a family of wavelets and scaling functions [8]–[12]. The approximation (scaled) signals are computed by filtering the signal using a low pass filter of length r, $h = [h_1, h_2, ..., h_r]$. On the other hand, the detail signals are computed by filtering the signal using a high pass filter of length r, $g = [g_1, g_2, ..., g_r]$. Finally, repeating the decomposing process on any scale J, the original signal can be represented as the sum of all detail coefficients and the last approximation coefficient.

Consider the following discrete signal $x(n)$ of length N where $N = 2^J$ for some integer J. At the first level of SWT, the input signal $x(n)$ is convolved with the $h_1(n)$ filter to obtain the approximation coefficients $a_1(n)$ and with the $g_1(n)$ filter to obtain the detail coefficients $d_1(n)$, so that:

$$a_1(n) = \sum_k h_1(n-k)x(k) \tag{1a}$$

$$d_1(n) = \sum_k g_1(n-k)x(k) \tag{1b}$$

because no sub-sampling is performed, $a_1(n)$ and $d_1(n)$ are of length N instead of $N/2$ as in the DWT case. At the next level of the SWT, $a_1(n)$ is split into two parts by using the same scheme, but with modified filters h_2 and g_2 obtained by dyadically up-sampling h_1 and g_1. Here the up-sampling operator inserts a zero between every adjacent pair of elements of $a_j(n)$.

The general process of the SWT is continued recursively for $j = 1, ..., J$ and is given as:

$$a_{j+1}(n) = \sum_k h_{j+1}(n-k)a_j(k) \tag{2a}$$

$$d_{j+1}(n) = \sum_k g_{j+1}(n-k)a_j(k) \tag{2b}$$

Therefore, the output of the SWT is then the approximation coefficients a_J and the detail coefficients $d_1, d_2, \ldots, d_J$, whereas the original signal $x(n)$ is represented as a superposition of the form:

$$x(n) = a_J(n) + \sum_{j=1}^{J} d_j(n) \tag{3}$$

The wavelet decomposition method is fully defined by the choice of a pair of low and high pass filters and the number of decomposition steps J. Hence, in this study we choose a pair of Haar wavelet filters given as:

$$h = \left[\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right] \tag{4a}$$

$$g = \left[\frac{-1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right] \tag{4b}$$

B. Proposed forecasting model

In order to predict the future signal $\hat{x}(n+h)$, our direct forecasting model will be the co-addition of two predicted values given as:

$$\hat{x}(n+h) = \hat{x}_a(n+h) + \hat{x}_{ia}(n+h) + e(n) \tag{5}$$

where $h = 3$ represents the forecasting horizon, $\hat{x}_a$ represents the annual cycle component, $\hat{x}_{ia}$ denotes the inter-annual

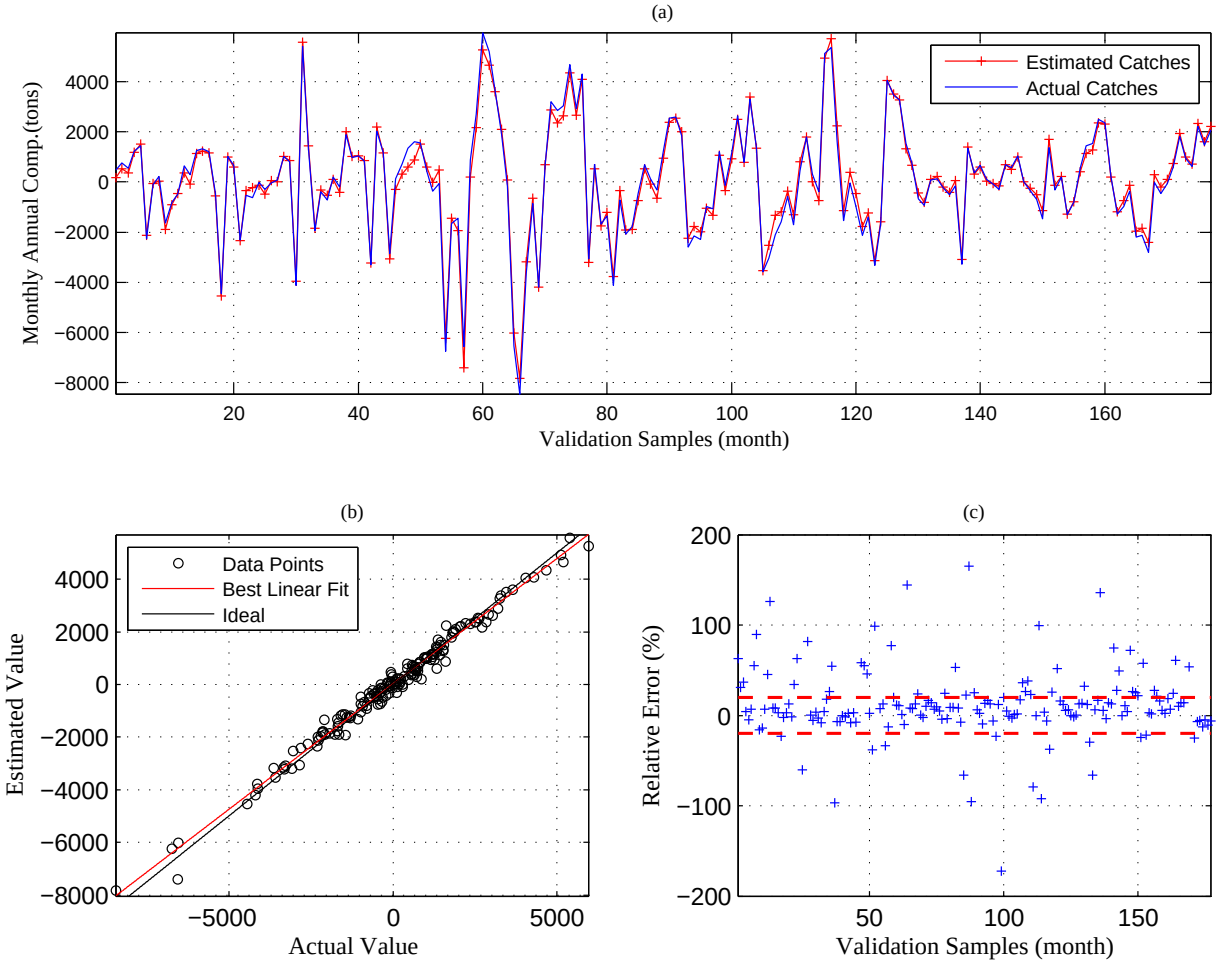

Fig. 2. Three-step-ahead forecasting for annual cycle component

component and e is an random process with distribution $\aleph(0, \sigma_e^2)$.

The components extraction process is described as follows and the flowchart is shows in Figure 1. The raw catches time series is decomposed using 3-level wavelet decomposition. The first detail component $d_1(n)$ of the direct output of the decomposition represents a (2-4)-month variability, the second detail component $d_2(n)$ of the direct output denotes a (4-8)-month dynamic, whereas the third detail component $d_3(n)$ contains some annual cycle as well as some large timescale variation with a (8-16) months dynamic. Therefore to extract annual cycle component $x_a(n)$, the detail components two and three of the direct output are combined $D = d_2 + d_3$ and then the new component D is subjected to 3-level wavelet decomposition.

The three detail components $d_1^{(2)}, d_2^{(2)}, d_3^{(2)}$ resulting from this wavelet decomposition are combined with the first detail component of the direct output d_1, whereas the third approximation component $a_3^{(2)}$ is combined with the third approximation component a_3 of the direct output to obtain the new inter-annual component x_{ia}.

The annual cycle component and the inter-annual component are estimated using a nonlinear autoregressive model with exogenous inputs given by the following equations

$$\hat{x}_a(n+h) = f([y_a(n), u_a(n)]) \tag{6}$$

where $y_a(n) = [x_a(n), x_a(n-1), ..., x_a(n-m)]$ are the endogenous inputs, $u_a(n) = [x_{ia}(n), x_{ia}(n-1), ..., x_{ia}(n-m)]$ are the exogenous inputs, m represents the memory of the model and $f(\cdot)$ is a nonlinear function.

$$\hat{x}_{ia}(n+h) = g([y_{ia}(n), u_{ia}(n)]) \tag{7}$$

where $y_{ia}(n) = [x_{ia}(n), x_{ia}(n-1), ..., x_{ia}(n-m)]$ are the endogenous inputs, $u_{ia}(n) = [x_a(n), x_a(n-1), ..., x_a(n-m)]$ are the exogenous inputs and $g(\cdot)$ is a nonlinear function.

In this paper the functions $f(\cdot)$ and $g(\cdot)$ are estimates using a RBF neural network (RBFNN) model.

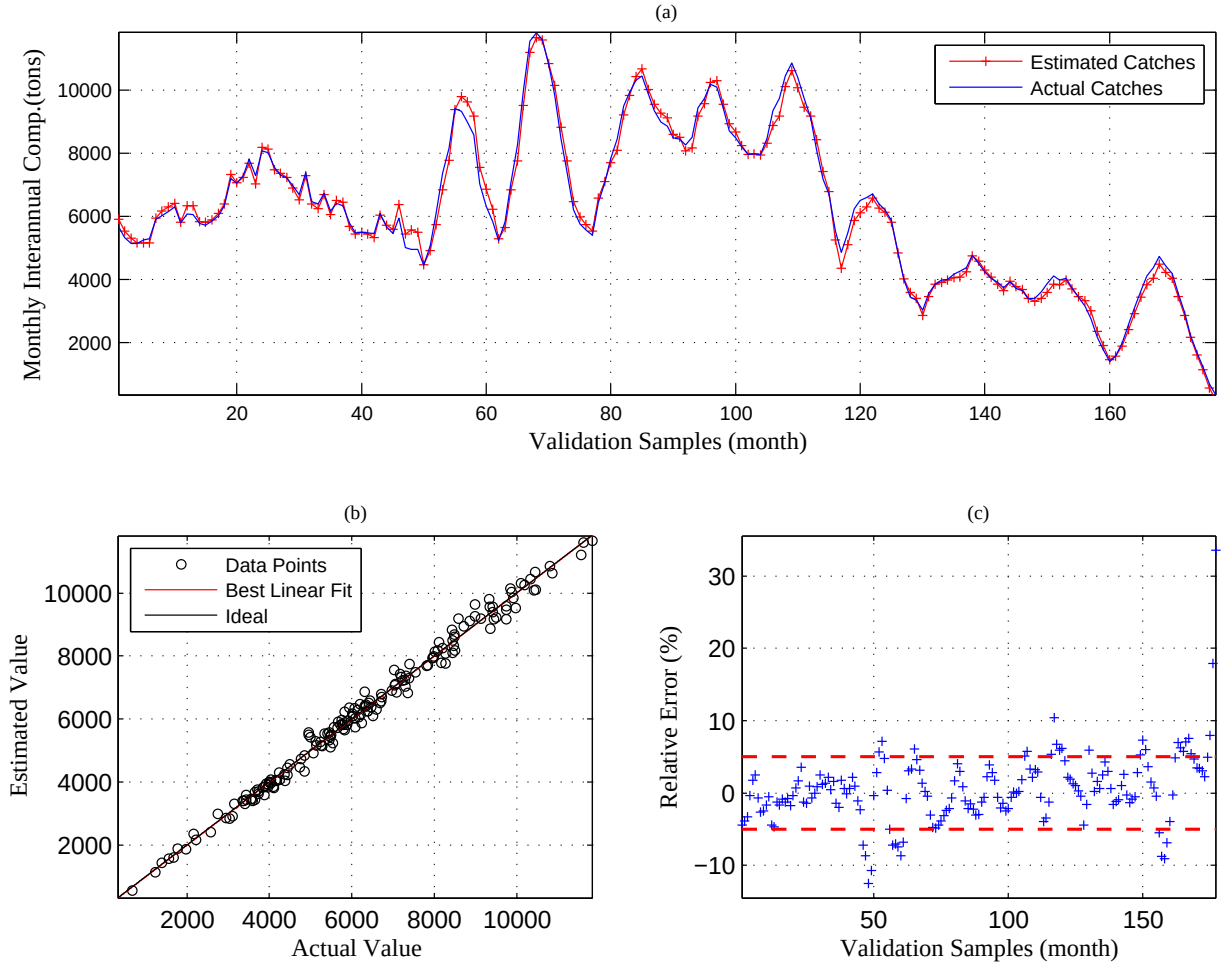

Fig. 3. Three-step-ahead forecasting for inter-annual component

C. RBF neural network model

The output of the RBFNN is obtained as

$$y = \sum_{j=1}^{N_h} b_j \phi_j(\| (z_i - v_{ji}) \|^2) \tag{8}$$

where N_h is the number of hidden nodes, z notes the regression vector containing $2m$ lagged values, $[b_1, \dots b_{N_h}]$ represents the linear output parameters, $v = [v_{j1}, v_2, \dots v_{j2m}]$ denotes the nonlinear parameters, and $\phi_j(\cdot)$ are hidden activation functions, which is given by:

$$\phi_j(\lambda) = \frac{1}{\sqrt{1+\lambda}} \tag{9}$$

In order to estimate the linear parameters and the nonlinear parameters of the RBFNN forecaster the separable nonlinear least squares algorithm is used, which is based on least square (LS) method [13] and Levenberg-Marquardt (LM) algorithms [14]. The LS algorithm is used to estimate the parameters b_j, whereas the LM algorithm is used to calibrate the nonlinear parameters v_{ji}. For any given representation of the nonlinear parameters, the optimal values of the linear parameters are obtained using the LS algorithm as follows:

$$b = \Phi^{\dagger} X \tag{10}$$

where X is the desired output patter vector and $\Phi^{\dagger}$ is the Moore-Penrose generalized inverse of the activation function output matrix Φ [13].

Once linear parameters are obtained, the LM algorithm adapts the nonlinear parameters of the hidden activation functions minimizing mean squared error. Finally, the LM algorithm adapts the nonlinear parameter $\theta = [b_j, v_{ji}]$ according to the following equations [14]:

$$\theta(n+1) = \theta(n) + \Delta\theta(n) \tag{11a}$$

$$\Delta\theta(n) = (\xi\xi^T + \mu I)^{-1}\xi^T e \tag{11b}$$

where ξ represents the Jacobian matrix of the error vector evaluated in θ and e_i is the error vector of the RBFNN for i-patter, I denotes the identity matrix and the parameter μ is increased or decreased at each step of the LM algorithm.

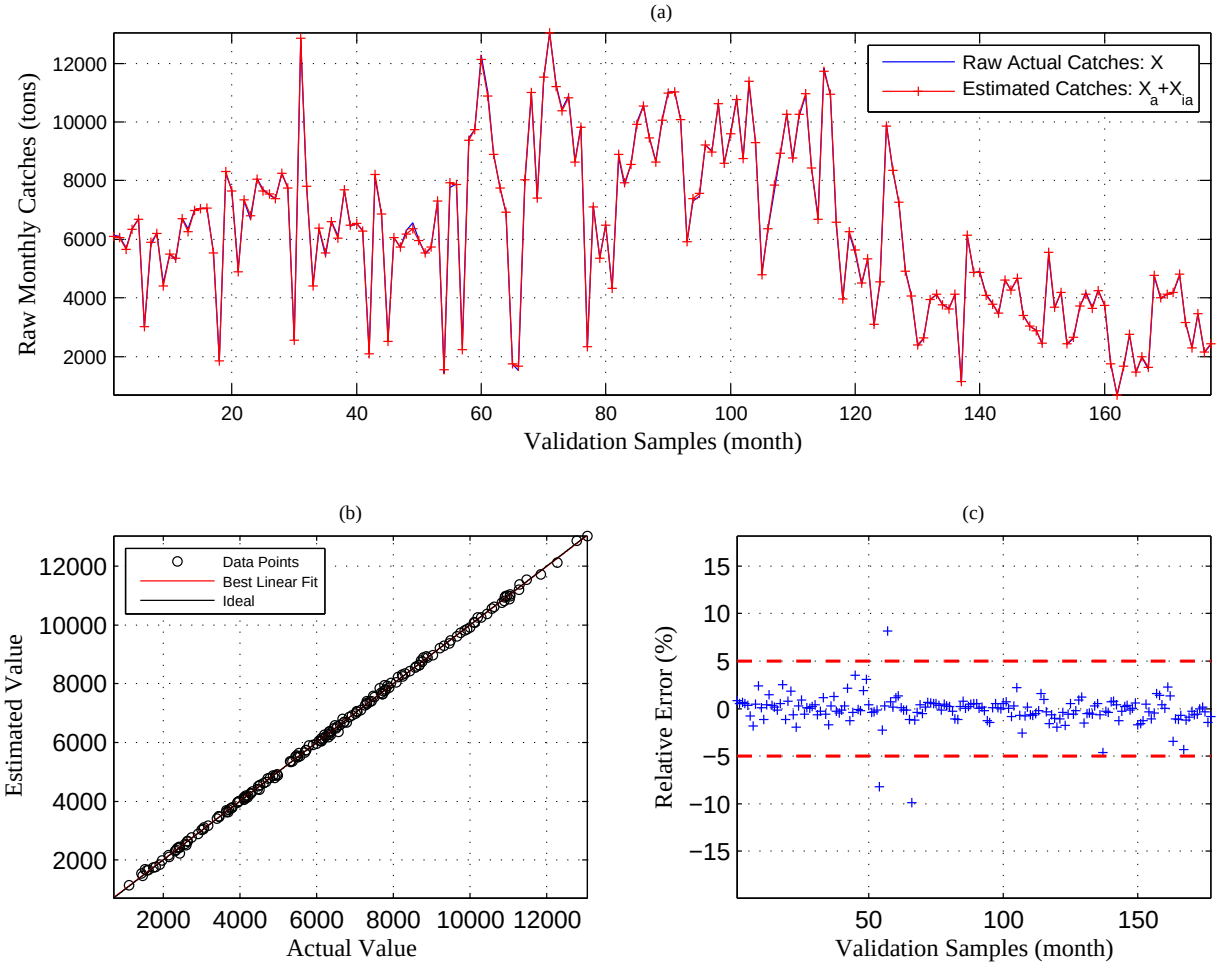

Fig. 4. Three-step-ahead forecasting for raw hake catches vs estimated hake catches

D. Performance Metrics

The forecasting accuracy is evaluated according to the root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE) and relative error (RE). The forecasting accuracy is better when the values of these measures are smaller. The definition of these metrics are given as follows:

$$RMSE = \sqrt{\frac{1}{M}\sum_{j=1}^{M}(x_i - \hat{x}_i)^2} \tag{12}$$

$$MAE = \frac{1}{M}\sum_{j=1}^{M}|(x_i - \hat{x}_i)| \tag{13}$$

$$MAPE = \frac{1}{M}\sum_{j=1}^{M}|(x_i - \hat{x}_i)/x_i| \tag{14}$$

$$RE = (x_i - \hat{x}_i)/x_i \tag{15}$$

where x_i is the actual value at time i, $\hat{x}_i$ is the forecasted value at time i and M is the number of testing samples.

III. Discussion

Total monthly hake catches off southern Chile were taken from Statistical Fishery Yearbooks (www.sernapesca.cl) for monthly period from January 1963 to December 2008 as shown in Figure 5(a), with a total of $N = 552$ observations. All raw monthly catches data set after subjected to the extraction process of components based on 3-level wavelet decomposition (illustrated in Figure 5), allowed to obtain the inter-annual cycle component and the annual component, which are shown in Figure 5(b) and Figure 5(c); respectively. Once the annual cycle and the inter-annual components were identified, the data set is divided into two parts: a training data set (T=370 observations) and a test data set (M=182 observations).

The training data is firstly used to choose the parameters of the RBFNN models, and the testing data set is used to compute the performance metrics of the models and for validation purposes. The RBFNN was calibrated with $2m = 22$ previous months as input data due to the annual cycle effect of monthly hake catches. Finding the optimal number of hidden nodes is

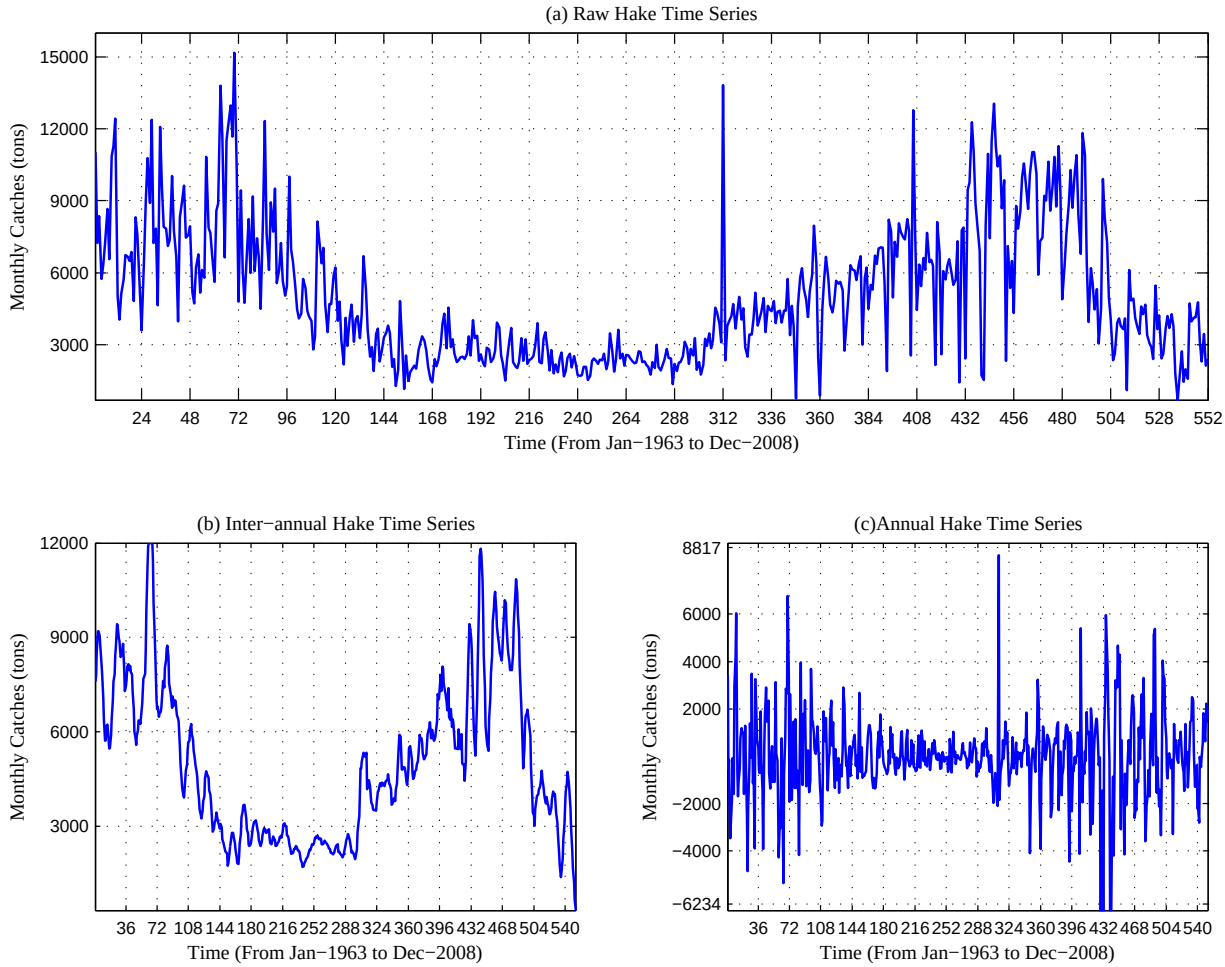

Fig. 5. Reconstruction of hake catches data derived from SWT

a complex problem, but in all our experiments the number of hidden nodes is set to $log(T)$. In the training process overall hidden weights were initialized with the training catches data and the stopping criterion was three iterations. After of the training process, the better architecture was calibrated with 22 input nodes, 6 hidden nodes and one output nodes and is denoted as RBFNN(22,6,1).

Now we present the 3-month-ahead forecasting results obtained with the RBFNN(22,6,1) model during the testing phase, whose results are illustrated in Figures 2, 3 and 4; respectively. Figure 2(a) provides observed annual cycle component versus forecasted annual cycle component, while that Figure 3(a) shows observed inter-annual component versus forecasted inter-annual component. On of other hand, Figures 2(b) and 3(b) show the regression curve between observed components and estimated components. From Figures 2(b) and 3(b) can be seen a good fit of the data to line $1:1$ with a 98% and 99% of the explained variance for annual component and inter-annual component; respectively.

Figure 4(a) provides raw monthly hake catches data versus forecasted hake catches, whose forecasting behavior is very accurate for testing data with a 99% of explained variance (Figure 4(b)), a MAE of 45 tons and a RMSE of 61 tons, while the explicated variance was of 99%. On the one hand, Figures 2(c), 3(c), and 4(c) depict relative error versus the predicted catches obtained by the RBFNN(22,6,1) model. It can be observed, that an important fraction of the catches tested are acceptable with residuals ranging from $\pm 5\%$.

IV. Conclusion

In this paper a 3-step-ahead forecasting strategy for monthly hake catches data set was proposed. The reason of the improvement in forecasting accuracy was due to use multi-scale stationary wavelet decomposition to separate both the annual and inter-annual components of the raw time series, since the behavior of each component is more smoothing than raw data set.

The forecasting strategy was applied to the monthly hake catches of the southern zone of Chile and the results show that 11 previous months of the annual component and 11 previous months of the inter-annual component contain valuable information to explicate a highest variance level.

Finally, the 3-step-ahead RBFNN(22,6,1) forecasting model can be suitable as a very promising methodology to any other marine species of the fisheries industry.

ACKNOWLEDGMENT

This research was partially supported by the Chilean National Science Fund through the project Fondecyt-Regular 1131105 and by the VRIEA of the Pontificia Universidad Católica de Valparaíso.

REFERENCES

[1] K. I. Stergiou and E. D. Christou, "Modelling and forecasting annual fisheries catches: comparison of regression, univariate and multivariate time series," *Fisheries Research*, vol. 25, pp. 105–138, 1996.

[2] K. I. Stergiou, "Prediction of the mullidae fishery in the eastern mediterranean 24 months in advance," *Fisheries Research*, vol. 9, issues 1, pp. 67–74, 1990.

[3] J. C. Gutirrez-Estrada, C. Silva, E. Yañez, N. Rodrguez, and I. Pulido-Calvo, "Monthly catch forecasting of anchovy engraulis ringens in the north area of Chile: Non-linear univariate approach," *Fisheries Research*, vol. 86, pp. 188–200, 2007.

[4] J. Gutirrez-Estrada, E. Yañez, I. Pulido, C. Silva, F. Plaza, and C. Borquez, "Pacific sardine (sardinops sagax, jenyns 1842) landings prediction: A neural network ecosystemic approach," *Fisheries Research*, vol. 100, pp. 116–125, 2009.

[5] E. Yañez, F. Plaza, J. Gutierez, N. Rodriguez, M. Barbieri, I. Pulido, and C. Borquez, "Anchovy (engraulis ringens) and sardine (sardinops sagax) abundance forecast of northern Chile: A multivariate ecosystemic neural network approach," *Progress in Oceanography*, vol. 87, pp. 242–250, 2010.

[6] O. Kisi, "Stream flow forecasting using neuro-wavelet technique," *Hydrological Processes*, vol. 22, issues 20, pp. 4142–4152, 2008.

[7] F. K. Nima A., "Day ahead price forecasting of electricity markets by a mixed data model and hybrid forecast method," *International Journal of Electrical Power & Energy Systems*, vol. 30, issue 9, pp. 533–546, 2008.

[8] Z. Bai-Ling, C. Richard, M. Jabri, D. Dersch, and B. Flower, "Multiresolution forecasting for futures trading using wavelet decompositions," *IEEE Transaction on neural networks*, vol. 12, pp. 765–775, 2001.

[9] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," in *Lecture Notes in Statistics: Wavelets and Statistics. Springer-Verlag*, 1995.

[10] G. Nason and B. Silverman, "The stationary wavelet transform and some statistical applications." in *In: Lecture Notes in Statistics: Wavelets and Statistics*, 1995.

[11] J. Pesquet, H. Krim, and H. Carfantan, "Time-invariant orthonormal wavelet representations," *IEEE transactions on signal processing*, vol. 44, pp. 1964–1970, 1996.

[12] D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*, C. U. Press, Ed. Cambridge, England, 2000.

[13] D. Serre, *Matrices: theory and applications*, Springer, Ed. New York, 2002.

[14] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11(2), pp. 431–441, 1963.

Supply Chain Management by Means of Simulation

Borja Ponte, David de la Fuente, Raúl Pino, Rafael Rosillo, and Isabel Fernández

***Abstract*—Several changes in the macro environment of the companies over the last two decades have meant that the competition is no longer constrained to the product itself, but the overall concept of supply chain. Under these circumstances, the supply chain management stands as a major concern for companies nowadays. One of the prime goals to be achieved is the reduction of the Bullwhip Effect, related to the amplification of the demand supported by the different levels, as they are further away from customer. It is a major cause of inefficiency in the supply chain. Thus, this paper presents the application of simulation techniques to the study of the Bullwhip Effect in comparison to modern alternatives such as the representation of the supply chain as a network of intelligent agents. We conclude that the supply chain simulation is a particularly interesting tool for performing sensitivity analyses in order to measure the impact of changes in a quantitative parameter on the generated Bullwhip Effect. By way of example, a sensitivity analysis for safety stock has been performed to assess the relationship between Bullwhip Effect and safety stock.**

***Index Terms*—Artificial Intelligence, bullwhip effect, simulation, supply chain management.**

I. Introduction

A supply chain encompasses all participants and processes involved in satisfying customer demands around some products. The supply chain management covers, therefore, activities related to provisioning, production and distribution of the product, all of them placed between an upper node, which we will call factory, and a lower node, which we call shop retailer.

Thus, we must consider two main flows along it: the materials flow, including the distribution of the product from the factory to the shop retailer (downstream flow) and the information flow, which refers to transferring orders from the customer by the remaining members (upstream flow).

Analyzing the supply chain, Forrester [1] noted that small changes in customer demand are highly amplified along the supply chain, leading it to larger variations in demand supported by the different levels, as they are further away from customer. This is called the Bullwhip Effect (or Forrester Effect), which, according to the subsequent research by Lee et al. [2], is due to four main causes: errors in the forecasts of demands, inadequate lot sizing, variations in product prices over time, and the rationing policy for fear of stock breakage.

There have been several changes in the last two decades in the macro environment of the companies that have set up a new business perspective. This has led to the perception that competition is no longer limited to the product itself, but what really competes is the overall concept of the supply chain. From this perspective, the production function is considered to have a strategic role as a source of competitive advantage, so that the practices related to the supply chain management now represent one of the main concerns of business.

In these circumstances, it is especially emphasized the importance of proper management of the supply chain regarding different objectives. One of them is undoubtedly reducing the Bullwhip Effect. In fact, Disney and Towill [3] demonstrated that the Bullwhip effect leads the supply chain to unnecessary costs that can represent, in some cases, more than 30% of the total costs thereof. That is to say, the Bullwhip Effect can be considered as one of the main causes of inefficiencies in supply chain management, which is produced by the conducts of the various players involved in it.

In this context, this work proposes the application of modern simulation techniques to the study of Bullwhip Effect in a supply chain.

The authors recently studied the subject through a multiagent approach in [4], and now, they proposes the use of simulation tools to complement the analysis. To develop the model, we have used the software ARENA 11.0. From this perspective, we will do simultaneously a comparative evaluation of the two alternatives, in order to assess the potential of each one and to open new ways of research of this problem.

The presented document is divided into four sections besides this introduction. Section 2 shows a review of the most relevant and recent literature on the subject. Section 3 describes the model which we have created, with the various elements that compose it. Section 4 presents the results of applying the model on different series. Finally, Section 5 presents the conclusions according to the planned objectives.

Manuscript received on July 24, 2013; accepted for publication on September 30, 2013.

Borja Ponte is a PhD student at the Polytechnic School of Engineering (University of Oviedo), Campus de Viesques s/n, CP 33204, Gijón (Asturias), Spain (e-mail: uo183377@uniovi.es).

David de la Fuente is with the Polytechnic School of Engineering (University of Oviedo), Campus de Viesques s/n, CP 33204, Gijón (Asturias), Spain (e-mail: david@uniovi.es).

Raúl Pino is with the Polytechnic School of Engineering (University of Oviedo), Campus de Viesques s/n, CP 33204, Gijón (Asturias), Spain (e-mail: pino@uniovi.es).

Rafael Rosillo is with the Polytechnic School of Engineering (University of Oviedo), Campus de Viesques s/n, CP 33204, Gijón (Asturias), Spain (e-mail: rosillo@uniovi.es).

Isabel Fernández is with the Polytechnic School of Engineering (University of Oviedo), Campus de Viesques s/n, CP 33204, Gijón (Asturias), Spain (e-mail: ifq@uniovi.es).

II. Background: Bullwhip Effect Reduction Through Simulation Tools

Firstly, we outline the traditional solutions proposed to mitigate the Bullwhip Effect. In the second and third, we briefly describe the most recent applications of multiagent methodology and simulation techniques to its reduction.

A. Traditional Solutions to the Bullwhip Effect.

Each supply chain has its own characteristics, mainly conditioned by the type of product which is offered to the consumer and by the market conditions in which it moves, and that unquestionably complicates the analysis of valid methodologies for reducing the Bullwhip Effect. However, it is possible to find some common problems to all of them, and several authors have proposed general strategies to be adapted to each particular supply chain. These traditional solutions to Bullwhip Effect are mainly based on collaboration among the various members of the supply chain, often sharing some information. Thus, some practices that are carried out in some companies and which have been successful in reducing the Bullwhip Effect are:

- Use of Information Technology systems, such as electronic data interchange [5].
- Postponement, which is based on a redesign of products with the aim that the differentiation takes place in nodes near the customer [6].
- Efficient Consumer Response (ECR). These are associations of companies to synchronize the supply chain [7].
- Vendor Managed Inventory (VMI). The supplier controls the inventory of the consumer, deciding on delivery times and quantities [8].
- Collaborative Planning, Forecasting and Replenishment (CPFR). It means that members of the supply chain can develop, in a collaborative way, business plans and processes [9].

B. Multiagent Systems in the Supply Chain Management.

The supply chain management is a highly complex problem, conditioned by multiple agents, each of which has to serve a large number of variables. In the last two decades, authors have looked for different ways to optimize the management by using new techniques based on Artificial Intelligence. Several authors have approached the supply chain as a network of intelligent agents. These are called multiagent systems.

Fox et al. [10] were pioneers in the proposal of the organization of the supply chain as a network of cooperating intelligent agents. In their work, each agent executes one or more functions of the supply chain, coordinating their actions with other agents. Later, Shen et al. [11] developed the tool MetaMorph II, which, through an agent-based architecture, integrates partners, suppliers and customers with a lead company through their respective mediators within a supply chain network via the Internet.

Kimbrough et al. [12] studied whether a structure based on agents could be valid for the supply chain management, and they reached the conclusion that the agents were able to effectively play the well known Beer Game [13], reducing the Bullwhip Effect. Moxaux et al. [14] used a multiagent system for modeling the behavior of each company in the supply chain. The paper proposes a variant of the Beer Game, which they called "Quebec Wood Supply Game".

Liang y Huang [15] developed, based on a multiagent architecture, a model which allowed predicting the order quantity in a supply chain with several nodes, where each one of them could use a different system of inventory. De la Fuente and Lozano [16] presented an application of Distributed Intelligence to reduce the Bullwhip Effect in a supply chain, based on a genetic algorithm. Zarandi et al. [17] introduced Fuzzy Logic in the analysis.

Wu et al. [18] applied the multiagent methodology to establish a supply chain model and to analyze in detail the Bullwhip Effect created along the chain, considering the non existence of information exchange among different members. One of the last studies in that regard is the one by Saberi et al. [19]. It develops a multiagent system, and which links the various agents that form it, emphasizing the collaborative aspect. Recently, Ponte and De la Fuente [4] proposed a multiagent model for managing the supply chain, based on collaboration between the various members, showing that this alternative allows a great reduction of Bullwhip Effect.

We can conclude that several changes in the last decades have become the supply chain in a complex system that requires modern methodologies for its analysis, seeking to optimize their management.

C. Supply Chain Simulation.

Digital simulation is a technique that allows imitating in a computer the behavior of a real system. Although it has been used for several decades, the continuous evolution of computers increases significantly its applications. It allows studying, in highly complex systems, the effect of small changes, which in real conditions it would not be feasible to analyze.

Manyem and Santos [20] simulated a supply chain of only two stages, with the aim of studying the propagation of Bullwhip Effect between two consecutive levels and, thus, the impact of this phenomenon in the profitability of the companies. They focused on the consequences of increasing the lead time, demonstrating that it introduced a great uncertainty to the chain, which meant severe disruption of performance.

Merkureyev et al. [21] simulated, using ARENA 5.0, a supply chain of four levels and described the impact on the Bullwhip Effect of the two supply chain information management policies (centralized and decentralized information) combined with two management policies inventory (policy min-max and stock-to-demand). Of these combinations, four different models emerged, concluding that the Bullwhip effect appeared in all cases, but not to the same extent, demonstrating that centralized information policy and the policy of stock-to-demand work best in Effect terms Bullwhip reduction.

Boute and Lambrecht [22] simulated the supply chain through a large spreadsheet, oriented to analyze the relationship between the level of customer service and the Bullwhip Effect. To do this, they studied various changes in the parameters that define replenishment policies at different levels. The authors concluded that the Bullwhip Effect can be reduced, even increasing fluctuations in inventories, but at the cost of lowering the level of customer service.

So, in contrast with multiagent systems, we refer in this section to simulations that do not introduce intelligence to the model so that agents do not have decision-making capacity in search of an optimal solution, but merely they follow a sequence of planned operations. We can conclude, after studying the literature, that these simulations are mainly used to analyze the consequences of the change of certain variables on the Bullwhip Effect, which also allows us to propose new solutions. We also noticed that there are not, compared to other techniques, many studies about the applications of supply chain simulation to analyze the Bullwhip Effect.

III. Model

To prepare the base model, we have considered a traditional supply chain with linear structure, which consists of five main levels: Consumer, Shop Retailer, Retailer, Wholesaler and Factory. Fig. 1 shows the graphical representation of the levels, indicating the materials flow, which occurs from the top of the chain (Factory) to the lower levels (Consumer). Therefore, it is called downstream flow. The information flow is considered to be in the opposite way, which is called downstream flow.

To implement the model, we have used ARENA 11.0 (developed by Rockwell Software).

The model is based on four Communication Channels (one between Consumer and Shop Retailer, other between Shop Retailer and Retailer, other between Retailer and Wholesaler, and the last between Wholesaler and Factory). Each one is similar to that shown in Fig. 2, which corresponds to a screenshot of the Channel Communication between the Shop

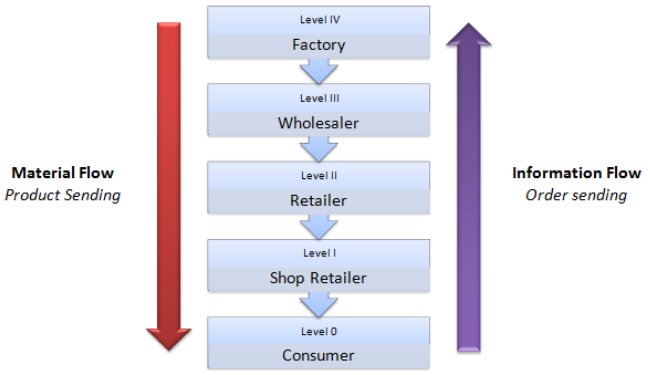

Fig. 1. Supply Chain Structure.

Retailer and the Retailer. In each one of them, the simulation begins with the level receives the order made by the previous one. Orders placed by level n+1 of the supply chain in period t ($O,t,n+1$) will mean the demand in the same period of the previous level (D,t,n), which is the main link connecting the different levels. It is expressed in (1).

$$D_t^n = O_t^{n+1} \quad (1)$$

Then, demand is stored in an external file for later analysis. Next, it evaluates whether the stock available at the level (IS,t,n), which has been planned according to the forecast of the demand, is sufficient to satisfy demand (D,t,n). If it is sufficient, it decreases the retailer's stock (FS,t,n) and it sends the material (Y,t,n). The stockout would be zero (SO,t,n). If it is not enough, stockout is generated in the level, which will be repaired in the next period, leaving the retailer's inventory to zero. In either case, it closes the communication between the two levels of the supply chain until the next period. Obviously, the initial stock of each level (IS,t,n) coincides with the sum of the final stock of this level in the previous period ($FS,t-1,n$) and the order made in the previous period ($O,t-1,n$). All this is expressed in (2), (3), (4), and (5).

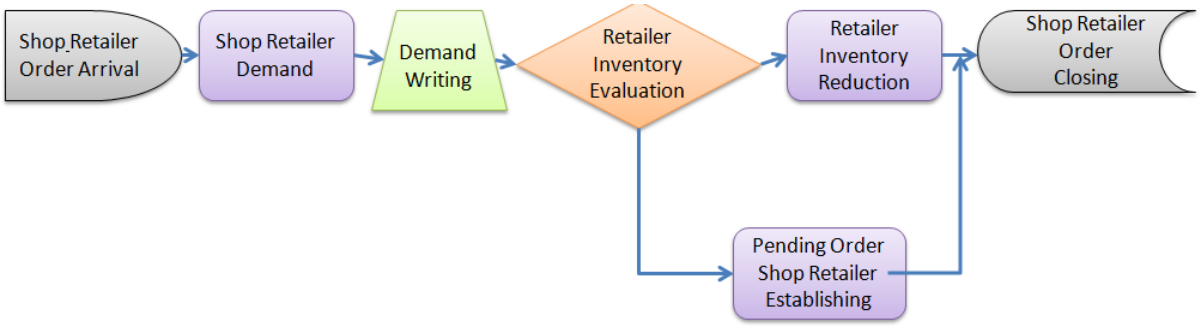

Fig. 2. Example of the Communication Channel between Shop Retailer and Retailer in the developed model.

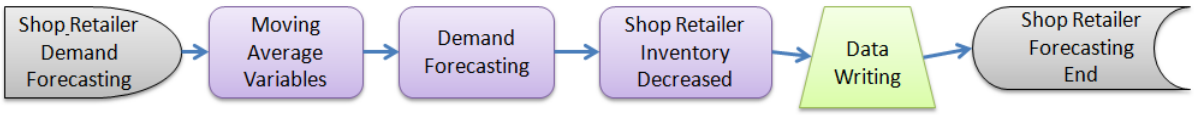

Fig. 3. Example of Forecasting Demand System of the Shop Retailer in the developed model.

$$Y_t^n = \min\{ D_t^n, IS_t^n \} + SO_{t-1}^n \quad (2)$$

$$FS_t^n = \min\{ IS_t^n - D_t^n, 0 \} \quad (3)$$

$$SO_t^n = \max\{ D_t^n - IS_t^n, 0 \} \quad (4)$$

$$IS_t^n = FS_{t-1}^n + O_{t-1}^n \quad (5)$$

Moreover, the model contains four Forecasting Demand Systems, one for each one of the four main levels of the supply chain. It is based on storing the last five demands (*D,t,n-i*) received from the previous level of the supply chain and, based on them, the estimation of demand in the next period (*FD,t,n*). Thereby, the technique for the demand forecasting is a moving average of five periods. We chose this method becasuse we will simulate the system with random time series, which are statistical distributions. It is not necessary to use more complex forecasting methods, oriented to time series with periodicity and tendency. We can express the forecasting method in (6).

$$FD_t^n = \frac{D_{t-1}^n + D_{t-2}^n + D_{t-3}^n + D_{t-4}^n + D_{t-5}^n}{5} \quad (6)$$

From there, the order which will be made by each level (*O,t,n*) considers the forecasting (*FD,t,n*), taking into account available stock (*IS,t,n*), stockout generated in the previous period (*SO,t-1,n*) and the safety stock which has decided the level (*SS,n*). It is expressed in (7). Furthermore, these data are also stored in an external file for later analysis. As an example, a screenshot of the Forecasting Demand Systems of the Retailer at a time of the simulation is shown in Fig. 3.

$$O_t^n = \max\{ FD_t^n - IS_t^n + SO_{t-1}^n + SS^n, 0 \} \quad (7)$$

IV. Results

Once developed the model in ARENA 11.0, we have conducted various tests on it, mainly with the aim to evaluate the advantages and disadvantages offered by this alternative, compared with the multiagent methodology [4]. Draw conclusions on the causes of the Bullwhip Effect is not the aim of this paper.

Section 5 includes this comparison, emphasizing the application of supply chain simulation techniques to perform sensitivity analysis to assess the effect of changes in system parameters on the Bullwhip Effect in the Supply Chain.

As an example, we detail the calculations performed to determine the impact of the security stock in the generation of Bullwhip Effect along a specified supply chain. For this, we have conducted five different simulations. In the first case, all levels of the supply chain are working without a safety stock. In the rest, the safety stock is 5 (low), 10 (middle), 20 (high) to 40 (very high). In all cases, it was considered that the demand follows a normal distribution with mean 100 and standard deviation 10. Furthermore, we have carried out, in the five different cases, a simulation of 1500 days (approximately 214 weeks), considering a training period of 100 days (14 weeks), where the results are not considered in the calculation of final values.

TABLE I:
Results of the Tests Related to the Safety Stock (I)

Variance	***SS=0***	***SS=5***	***SS=10***	***SS=20***	***SS=40***
Consumer	116.83	116.83	116.83	116.83	116.83
Shop Retailer	170.42	170.19	170.20	170.91	175.15
Retailer	261.19	261.01	261.98	267.33	291.72
Wholesaler	418.24	418.61	422.19	438.98	511.06
Factory	694.65	696.47	705.60	745.81	914.00

TABLE II:
Results of the Tests Related to the Safety Stock (II)

Bullwhip Effect	***SS=0***	***SS=5***	***SS=10***	***SS=20***	***SS=40***
Shop Retailer	1.459	1.457	1.457	1.463	1.499
Retailer	1.533	1.534	1.539	1.564	1.666
Wholesaler	1.601	1.604	1.612	1.642	1.752
Factory	1.661	1.664	1.671	1.699	1.788
Supply Chain	***5.946***	***5.961***	***6.040***	***6.384***	***7.823***
Increase over SS=0	–	0.25%	1.56%	7.37%	31.57%

Table I shows the results of the five tests. It contains the variance of orders placed by each member of the supply chain throughout the simulation period. Note that the variance of the factory refers to the production rate, since it is the highest level of the chain.

Many authors quantify the Bullwhip Effect in supply chain as follows:

$$BW = \frac{\sigma_{df}^2}{\sigma_{dc}^2} \quad (8)$$

where σ_{dc}^2 is the variance in consumer demand for the product, and σ_{df}^2 represents the variance in the rate of the factory production.

Likewise, the Bullwhip Effect generated at each step can be defined as the ratio of the variance in orders sent to the next level of the supply chain (σ_{out}^2), and the variance in orders received from the previous level of the supply chain (σ_{in}^2). It is expressed in (9).

$$BW_i = \frac{\sigma_{out}^2}{\sigma_{in}^2} \quad (9)$$

This allows expressing the Bullwhip Effect along the entire supply chain as the product of the ratios that define the Bullwhip Effect at each level.

$$BW = \prod_{i=1}^{n} BW_i \quad (10)$$

Thereby, Table II contains the results of the test, but oriented Bullwhip Effect generation. It contains both the Bullwhip Effect generated at different levels, by (9), and the overall Bullwhip Effect generated in the supply chain in each case, by (10). Besides, it includes in the four cases in which it works with safety stock, the increase in the Bullwhip Effect related to the case with no safety stock.

Fig. 4. Relationship between safety stock and the Bullwhip Effect.

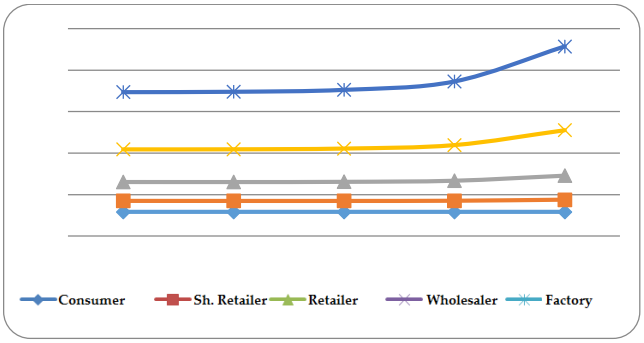

Fig. 5. Variance of the various levels in the different cases of safety stock.

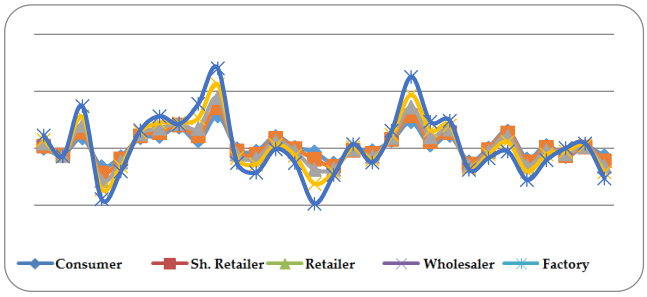

Fig. 6. Evolution in the demands from week 171 to week 200 in the simulation with 40 units of Safety Stock in each level.

Tables I and II show the relation in the studied time serie, between the safety stock and the Bullwhip Effect generated in the supply chain.

First, it is possible to note that in all cases the amplification of the variance of the demand for each level is increased as it moves away from the consumer. Moreover, the consideration of safety stock, in order to decrease the probability of stockout, increases the amplification of the variability of demands. That is, it is possible to conclude that in this case the Bullwhip Effect is greater when safety stock is increased. However, this increase is small, relatively speaking, while the safety stock used is not high.

Thereby, in this case, for safety stock levels which may be called "reasonable", its impact on the Bullwhip Effect is low. From there, it rises considerably, as it is possible to see in Fig. 4, which seems to outline an exponential relationship between the two variables.

Figure 5 shows the evolution of the variance of the demand of the various levels of the supply chain in the five different simulations throughout the simulation period (data obtained from Table I). Finally, Fig. 6 represents, as an example, variation of the demands in the different levels of the supply chain for thirty intermediate weeks (from 171 to 200) in the simulation related to 40 units of safety stock.

Finally, we want to recall that the purpose of this paper is not to assess the effect of stock safety in the Bullwhip Effect generation (it has only been done as an example for a particular time series and without the aim of drawing conclusions), but to evaluate the application of supply chain simulation techniques in the analysis of the Bullwhip Effect.

V. Conclusion: Comparative Analysis of Tools

This work complements the study of the Bullwhip Effect through multiagent methodology conducted by the same authors recently [4]. Having developed and implemented both tools based on a similar model, and after analyzing them with sufficient detail, we can conclude that:

- The approach to the problem through a multiagent system is a suitable alternative for the development of intelligent software, which is able to decide, based on a set of solutions available, the optimum for the system, according to selected criteria (related to improve the supply chain management). Through coordination between the various agents which simulate each level, this option allows to split a highly complex problem into a set of smaller problems. It is also possible to introduce advanced forecasting methods, offering a high performance in reducing Bullwhip Effect. Thus, the multiagent methodology improves supply chain management through collaboration between the different levels of it.
- The multiagent system can also analyze the causes of the Bullwhip Effect, especially the main one, related to errors in demand forecasts. However, sensitivity analysis, based on the study of the impact of a particular quantitative parameter on the Bullwhip Effect, simulation is a more appropriate tool. It allows this study with a high ease and efficiency. Through the supply chain simulation, it is affordable to simulate a time period large enough in different conditions, which it allows to decide which is the best alternative. The risk assumed with this alternative is minimal compared to the evaluation of various alternatives in the real supply chain. From there, it is possible to achieve a better understanding of the phenomenon, which may help to mitigate their appearance in order to reduce their harmful impact on the supply chain.

Acknowledgments

This work was supported by the Government of the Principality of Asturias, through the “Severo Ochoa” Predoctoral Grants for Research and Teaching of the Principality of Asturias.

REFERENCES

[1] J.W. Forrester, "Industrial dynamics", MIT Press. Cambridge, MA, 1961.

[2] H.L. Lee, V. Padmanabhan, and S. Whang, "The bullwhip effect in supply chains", *Sloan Management Review*. Vol. 38(3), pp. 93–102, 1997.

[3] S.M. Disney, and D.R. Towill, "The effect of Vendor Managed Inventory (VMI) dynamics on the Bullwhip effect in supply chain", *International Journal of Production Economics.* Vol. 85, pp. 199–215, 2003.

[4] B. Ponte, and D. De la Fuente, "Multiagent System to Reduce the Bullwhip Effect", *Proceedings of the 2013 International Conference on Agents and Artificial Intelligence*. Vol. 1, pp. 67–76, 2013.

[5] J.A. Machuca, and R. Barajas "The impact of electronic data interchange on reducing bullwhip effect and supply chain inventory cost", *Transportation Research Part E*. Vol. 40, pp. 209–228, 2004.

[6] L. Chen, and L. Lee Hau, "Information Sharing and Order Variability Control Under a Generalized Demand Model", *Management Science*. Vol. 55(5), pp. 781–797, 2009.

[7] S.M. Disney, and D.R. Towill, "Transfer function analysis of forecasting induced bullwhip in supply chain", *International Journal of Production Economics*. Vol. 78, pp. 133–144, 2002.

[8] J. Holmstrom, "Product range management: a case study of supply chain operations in the European grocery industry", *Supply Chain Management*. Vol. 2(3), pp. 107–115, 1997.

[9] Y.F. Ji, and H.L. Yang, "Bullwhip Effect Elimination in Supply Chain with CPFR", *Proceedings of the 2005 International Conference on Management Science & Engineering*. Vol 1–3, pp. 737–740, 2005.

[10] M.S. Fox, J.F. Chionglo, and M. Barbuceanu, "The Integrated Supply Chain Management System", Internal Report, Univ. of Toronto, 1993.

[11] W. Shen, D. Xue, and D.H. Norrie, "An Agent-Based Manufacturing Enterprise Infrastructure for Distributed Integrated Intelligent Manufacturing Systems", *Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Systems PAAM'98*, London, 1998.

[12] S.O. Kimbrough, D.J. Wu, and F. Zhong, "Computer the beer game: can artificial manage supply chains?" *Decision Support Systems.* Vol. 33, pp. 323–333, 2002.

[13] J.D. Sterman, "Modelling managerial behaviour: Misperceptions of feedback in a dynamic decision making experiment". *Management Science*. Vol. 35(3), pp. 321–339, 1989.

[14] T. Moyaux, B. Chaib-draa, and S. D'Amours, "An agent simulation model for the Quebec forest supply chain". *Lecture Notes in Artificial Intelligence.* Vol. 3191, pp.226–241, 2004.

[15] W.Y. Liang, and C.C. Huang, "Agent-based demand forecast in multi-echelon supply chain". *Decision Support Systems.* Vol. 42(1), pp. 390–407, 2006.

[16] D. De la Fuente, and J. Lozano, "Application of distributed intelligence to reduce the bullwhip effect". *International Journal of Production Research.* Vol. 44(8), pp. 1815–1833, 2007.

[17] M.H. Zarandi, M. Pourakbar, and I.B. Turksen, "A fuzzy agent-based model for reduction of bullwhip effect in supply chain systems", *Expert systems with applications*. Vol. 34(3), pp. 1680–1691, 2008.

[18] S.N. Wu, W.H. Gan, and F.M. Wei, "Analysis on the Bullwhip Effect Based on ABMS", *Procedia Engineering*. Vol. 15, 2011.

[19] S. Saberi, A.S. Nookabadi, and S.R. Hejazi, "Applying Agent-Based System and Negotiation Mechanism in Improvement of Inventory Management and Customer Order Fulfilment in Multi Echelon Supply Chain", *Arabian Journal for Science and Engineering*. Vol. 37(3), pp. 851–861, 2012.

[20] P. Manyem, and D. Santos, "Factors that Influence the Bullwhip Effect and its Impact on Profitability: A Simulation Study of Supply Chains", *Proceedings of the 4th Annual International Conference on Industrial Engineering Theory*, Applications and Practice. San Antonio, 1999.

[21] Y. Merkureyev, J. Petuhova, and M. Buikis, "Simulation based statistical analysis of the bullwhip effect in supply chains", *18th European Simulation Multiconference Networked Simulations and Simulaed Networks ESM 2004*, pp. 13–16, 2004.

[22] R. Boute, and M. Lambrecht, "Exploring the Bullwhip Effect by Means of Spreadsheet Simulation". Open access publications form Katholieke Universiteit Leuven, 2009.

A POS Tagger for Social Media Texts Trained on Web Comments

Melanie Neunerdt, Michael Reyer, and Rudolf Mathar

***Abstract*—Using social media tools such as blogs and forums have become more and more popular in recent years. Hence, a huge collection of social media texts from different communities is available for accessing user opinions, e.g., for marketing studies or acceptance research. Typically, methods from *Natural Language Processing* are applied to social media texts to automatically recognize user opinions. A fundamental component of the linguistic pipeline in *Natural Language Processing* is *Part-of-Speech* tagging. Most state-of-the-art *Part-of-Speech* taggers are trained on newspaper corpora, which differ in many ways from non-standardized social media text. Hence, applying common taggers to such texts results in performance degradation. In this paper, we present extensions to a basic Markov model tagger for the annotation of social media texts. Considering the German standard *Stuttgart/Tübinger TagSet* (STTS), we distinguish 54 tag classes. Applying our approach improves the tagging accuracy for social media texts considerably, when we train our model on a combination of annotated texts from newspapers and Web comments.**

***Index Terms*—Natural language processing, part-of-speech tagging, opinion mining, German.**

I. Introduction

SOCIAL media applications lead to constantly growing user generated content in the Web. Different types of social media tools, e.g., blogs, forums as well as news sites allow users to post Web comments. This kind of consumer-to-consumer communication can efficiently be used to access user opinions for marketing studies or acceptance research. A beneficial property of Web comments is the fact that the data is natural and authentic, and public. Furthermore, opinions from proponents, opponents as well as from impartial persons can be obtained from different Web domains, i.e., different communities.

Besides automatic opinion recognition, many other *Natural Language Processing* (NLP) methods such as syntactical parsing, machine translation or text summarization require *Part-of-Speech* (POS) tag information for a given word sequence. State-of-the-art POS taggers are basically developed for the task of tagging standardized texts such as newspaper articles which are grammatically approved. Hence, parameter estimation is usually performed on newspaper text corpora as training data. Web comments, however, differ from standardized text, since they are characterized by a spoken language, a dialogic and an informal writing style. This poses some special challenges to deal with in developing methods for automatic POS tagging of Web comments. These are particularly, the treatment of unknown (out-of-vocabulary) words and the different grammatical structure of social media texts in contrast to newspaper text. Furthermore, text genre specific manually annotated corpora, i.e., Web comments are required for training and testing. To the best of our knowledge all large manually annotated corpora are exclusively newspaper texts.

In this work, we propose a Markov model tagger with parameter estimation enhancements for the POS annotation of social media texts. We apply and evaluate the tagger for German social media texts exemplarily. In order to make our method usable for NLP methods requiring POS information, e.g., syntactical parsing, we use the 54 *Stuttgart/Tübinger TagSet* (STTS) tag classes without any text genre specific extensions. Web comments are not completely different from newspaper texts. However, due to the dialogic text style, i.e. the distribution of POS sequences changes, the grammatical structure differs. Hence, the training is performed on a combination of newspaper and Web comment corpora. Results are compared to state-of-the-art/widely used POS taggers. We particularly study the influence of additional Web comment training data for our approach and compare results to those achieved by methods basically developed for standardized texts. The proposed approach outperforms state-of-the-art POS taggers significantly for German social media texts.

The outline of this paper is as follows: Section II summarizes the related work to provide an overview of POS tagging. In Section III we introduce *WebTagger* for automatic POS tagging applied to social media texts. Section IV presents experimental results considering different aspects, particularly discussing the adaptability to other languages in Subsection IV-E. Section V covers the conclusion and discusses future work.

II. Related work

Different statistical approaches have been proposed to solve the task of automatic POS tagging. Typically, POS taggers utilize two different probabilistic models, a Maximum entropy model or a Markov model capturing lexical and contextual information. Common Maximum entropy based taggers are proposed in [1], [2]. These approaches are adapted by using

Manuscript received on August 2, 2013; accepted for publication on September 30, 2013.

The authors are with the Institute for Theoretical Information Technology, RWTH Aachen University, Germany (e-mail: {neunerdt, reyer, mathar}@ti.rwth-aachen.de).

different features for the model. Toutanova et al. [1] propose the Stanford tagger modeling the sequence of words as bi-directional dependency network considering lexical and tag context information. Markov model taggers are proposed in [3], [4], [5]. TreeTagger [4] and TnT [5] use a second order Markov model applying some smoothing techniques for the estimation of lexical probabilities. The Stanford tagger and the TreeTagger are trained on corpora in different languages, which shows their generality in application. Both taggers use the STTS [6] tag set for German, which is commonly used for NLP methods. Furthermore, some other machine learning techniques, e.g., Support Vector Machines [7] and Neural Networks [8], are applied to the problem of automatic POS tagging.

In [9], [10] different POS taggers are compared and evaluated for German. Schneider et al. [9] point out the performance loss on unknown words, of a rule-based tagger compared to a statistical tagger. Five state-of-the-art taggers applied to Web texts are studied in [10]. Accuracy drops significantly for different Web text genres. Hence, the automatic annotation of Web texts is not yet a solved task.

Gimpel et al. [11], [12], [13] address the task of tagging non-standardized texts, characterized by a high number of unknown words. Gadde et al. [2] introduce adaptions to the Stanford tagger to handle noisy English text. Results are evaluated based on an SMS dataset. In [14] a twitter tagger based on a conditional random field (CRF) with features adapted to twitter characteristics is proposed. They propose some additional word clustering and further improvement to their method in [13].

III. WebTagger

WebTagger has much in common with the TreeTagger [4]. The taggers differ in the way the lexical probabilities are estimated, in particular for unknown words. We use the STTS tagset for annotation. Annotation rules for social media characteristics are given in [15], [16] and [17]. The general tagger model is taken from [4] and explained in Subsection III-A. Subsection III-B comprises our proposed enhancements to the basic tagger to improve POS tagging performance for social media texts. In contrast to other approaches estimation of lexical probabilities is extended by mapping unknown tokens (words), to tokens known from training or to some regular expressions. This mapping improves the estimation of lexical probabilities. Furthermore, prefix and suffix lexicon information are efficiently combined. Finally, in contrast to standard methods we suggest to use a semi-supervised auxiliary lexicon instead of information from automatically tagged or unsupervised training data.

A. Model

As tagger model we use an enhanced standard Markov model. In this subsection we explain the basic model. The aim of the tagger is to predict the associated POS tag sequence $t_1, \dots, t_n, \dots, t_N$ with $t_n \in \mathcal{T}$ (STTS) for a given sequence of tokens $w_1, \dots, w_n, \dots, w_N$ with $w_n \in \mathcal{W}$, where $\mathcal{W}$ contains all possible tokens. In order to achieve that the optimization problem

$$\hat{t}_1^N = \arg\max_{t_1^N} P(t_1^N, w_1^N)$$

using the notation for a sequence of POS tags t_l^n

$$t_l^n = \begin{cases} (t_l, \dots, t_n) & 1 \le l \le n \le N \\ (t_1, \dots, t_n) & l \le 0 \end{cases}$$

with $l \in \mathbb{Z}$, $n \in \mathbb{N}$, and $l \le n \le N$ is solved. The sequence of tokens w_l^n is defined analogously. This is a huge optimization problem which is simplified by the following approach. First, the probability chain rule for w_N and t_N to describe the joint probability by conditional probabilities is applied:

$$\begin{aligned} &P(w_1^N, t_1^N) = \\ &\quad P(w_N \mid w_1{}^{N-1}, t_1^N) P(t_N \mid w_1^{N-1}, t_1^{N-1}) P(w_1^{N-1}, t_1^{N-1}). \end{aligned} \tag{1}$$

As in [4] we use the assumptions

$$\begin{aligned} P(w_n \mid w_1^{n-1}, t_1^n) &= P(w_n \mid t_n), \\ P(t_n \mid w_1^{n-1}, t_1^{n-1}) &= P(t_n \mid t_{n-k}^{n-1}) \end{aligned}$$

with $k \in \mathbb{N}$. Applying those assumptions, a simple law of conditional probability, and iterating the procedure described in (1) leads to the equation:

$$P(w_1^N, t_1^N) = \prod_{n=1}^{N} \underbrace{\frac{P(t_n \mid w_n)}{P(t_n)}}_{\text{Lexical Prob.}} p(w_n) \underbrace{P(t_n \mid t_{n-k}^{n-1})}_{\text{Transition Prob.}}.$$

The assumptions are also referred to as k-order Markov model for transition probabilities and zero-order Markov model for the lexical probabilities. Moreover, the token probabilities $p(w_n)$ do not change with the tag sequences, and hence, may be omitted. Overall, this allows to model transition and lexical probabilities independently and the optimization task is rephrased as

$$\hat{t}_1^N = \arg\max_{t_1^N} \left\{ \prod_{n=1}^{N} \frac{P(t_n \mid w_n)}{P(t_n)} P(t_n \mid t_{n-k}^{n-1}) \right\}.$$

Before this optimization problem can be solved those probabilities have to be determined. The estimation of transition probabilities is taken from TreeTagger [4] by applying the ID3 algorithm [18]. Due to the ungrammaticalities, particularly given in social media texts, a high number of unseen contexts, e.g., trigrams, might occur when applying the tagger. In order to get reliable estimates in such cases, zero probabilities are replaced by a predefined small value. Furthermore, we propose to use manually annotated social media texts as additional training data, in order to learn different tag contexts given by the dialogic style characteristics of such texts. Lexical probabilities are estimated by our proposed methods, described in the following Subsection III-B.

B. Enhancements

In this subsection our enhancements to a basic Markov model tagger are introduced. This comprises the estimation of lexical probabilities, particularly for unknown tokens (words). Considering the high frequency of unknown tokens and their variability of POS classes makes the task of probability estimation more complex compared to newspaper texts. Hence, this problem can not be solved adequately by standard methods and more sophisticated methods are needed.

1) Token preprocessing: The given sequence of tokens usually contains tokens which do not occur in the training set. The preprocessing aims at mapping these tokens to related known tokens, if there exists a fitting token. Related tokens can be obtained by some transformations steps described by $t(w_n)$. These steps contain cross-language transformations as well as some transformations specifically for German. Amongst others character iteration correction, e.g., Hellooooooo→Hello, or *Umlaut* correction, e.g., Huette→Hütte (cottage), or character correction, e.g., Kuss→Kuß (kiss). Such transformations may be interpreted as substituting tokens by its normalized version. Therefore, we are calling this kind of transformation *normalization*. Furthermore, there are language independent *word classes* which are easily recognized and anticipated using regular expressions. Some examples are emoticons, e.g., *:-)* and *:(*, and URLs, e.g., http://www.test.de, xy.ch, multiple punctuation marks, e.g., and ....., *!!!*, number replacements, e.g., *50er*. The set of possible POS tags for each word class differs from one to three. In summary, our preprocessing step substitutes unknown tokens by its transformation, if this is within the training set, or returns the regular expression r, if the word is described by it or returns the marker for unknown tokens ϵ. This procedure is described by the mapping function $m : \mathcal{W} \to \mathcal{X} \cup \{\epsilon\}$ which is defined as

$$m(w_n) = \begin{cases} /r/ & w_n \in \mathcal{W}_r, \\ w_n & w_n \in \mathcal{L} \setminus \mathcal{R} \\ t(w_n) & t(w_n) \in \mathcal{L} \setminus \mathcal{R} \wedge w_n \notin \mathcal{L} \setminus \mathcal{R} \\ \epsilon & \text{elsewise.} \end{cases} \quad (2)$$

An overview of the corresponding word sets is given in Table I.

TABLE I
WORD SETS.

Word Set	Description
$\mathcal{R}$	Tokens covered by regular expressions
$\mathcal{L}$	Full form lexicon created from training data
$\mathcal{X} = \mathcal{L} \cup \mathcal{R}$	Full form lexicon extended by regular expressions
$\mathcal{W}$	All possible tokens

The word set $\mathcal{X}$ contains all tokens given by the full form lexicon $\mathcal{L}$ created from supervised training data, extended by the set of words $\mathcal{R}$ created by all regular expressions $r \in R$ as follows:

$$\mathcal{W}_r = \{w \in \mathcal{W} \mid w \sim /r/\}$$

indicates all tokens covered by a regular expression $r \in R$, thus $\mathcal{R} = \dot{\bigcup}_{r \in R} \mathcal{W}_r$.

2) Parameter estimation: Before our tagger can be used for predicting POS tag sequences, the probability parameter values have to be estimated. Therefore, we basically use a supervised learning approach, but extend this by some semi-supervised learning technique which is explained in more detail in the following section. A manually annotated training corpus

$$\mathcal{TR} = \left\{(\hat{w}_n, \hat{t}_n) \mid 1 \leq n \leq N\right\}$$

is used, where for each word $\hat{w}_n$ the correct tag $\hat{t}_n$ is known. The lexicon is given as

$$\mathcal{L} = \{\hat{w}_n \mid 1 \leq n \leq N\}.$$

We assume lexical probabilities to be position independent. Hence, we replace $P(t_n \mid w_n) = P(t \mid w)$ in the following notation. If the word $m(w)$ is known, i.e., it occurs in the training set $\mathcal{TR}$, the estimation is given by

$$\hat{P}_L(t \mid w) = \frac{\mid \left\{k \mid (\hat{t}_k, \hat{w}_k) = (t, w)\right\} \mid}{\mid \{k \mid \hat{w}_k = w\} \mid},$$

where the index L indicates that the word w is in the lexicon $\mathcal{L}$.

In the following the estimates if the word $m(w)$ is not in the lexicon $\mathcal{L}$ are described. First, we explain the estimation of the probabilities, if the unknown word is represented by a regular expression. The probabilities are given as

$$\hat{P}_R(t \mid r) = \frac{\mid \left\{k \mid \hat{t}_k = t \wedge \hat{w}_k \in \mathcal{W}_r)\right\} \mid}{\mid \{k \mid \hat{w}_k \in \mathcal{W}_r\} \mid}. \quad (3)$$

Using these estimates for regular expressions enables to assign reliable tag distributions even to previously unseen tokens from training.

Now we deal with (still) unknown tokens. From the training data set we determine all prefixes and suffixes of maximal length five. The description of all tokens having the same prefix/suffix may be realized with a regular expression. Hence, we assess the lexical probabilities for all given prefixes and suffixes as in (3). However, to improve the quality of our estimation we combine the probabilities for prefixes and suffixes as follows:

$$\hat{P}_{PS}(t \mid w) = \frac{\mid \left\{k \mid \hat{t}_k = t \wedge \hat{w}_k \in \mathcal{W}_{p(w)})\right\} \mid + \mid \left\{k \mid \hat{t}_k = t \wedge \hat{w}_k \in \mathcal{W}_{s(w)})\right\} \mid}{\mid \left\{k \mid \hat{w}_k \in \mathcal{W}_{p(w)}\right\} \mid + \mid \left\{k \mid \hat{w}_k \in \mathcal{W}_{s(w)}\right\} \mid}$$

where $p(w)/s(w)$ are the regular expressions for the prefix/suffix of the word w. Common approaches use the joint probabilities of the independent prefix and suffix distributions. However, combining prefix and suffix lexical probabilities by the arithmetic mean, makes the method robust for uncommon prefix and/or suffix, which arise from informal writing style characteristics, e.g., word shortenings or typing errors. The proposed method improves tagging accuracy by 0.5 percentage points compared to the commonly used joint probability

approach. In summary, the lexical probability is given as

$$P(t \mid w) = \begin{cases} \hat{P}_L(t \mid m(w)) & m(w) \in \mathcal{L}, \\ \hat{P}_R(t \mid m(w)) & m(w) \in \mathcal{R} \setminus \mathcal{L}, \\ \hat{P}_{PS}(t \mid w) & w \in (\mathcal{P} \cup \mathcal{S}) \setminus \mathcal{X}, \\ \hat{P}_S(t \mid w) & \text{elsewise}, \end{cases} \quad (4)$$

where $\mathcal{P}/\mathcal{S}$ describes all tokens with known prefixes (suffixes). The last case in the description is by default given by the frequencies of the tags in the training set

$$\hat{P}_S(t \mid w) = \frac{|\{k \mid \hat{t}_k = t\}|}{N},$$

which is independent of the word w.

3) Semi-supervised learning: Preparing a fully supervised training text is a time-consuming job. In this subsection we propose an alternative approach which reduces the annotation effort considerably. The basic idea is as follows. The tagger is used for automatic tagging of a large social media text corpus.

$$\mathcal{SL} = \{(w_m, t_m) \mid 1 \leq m \leq M\}$$

The most frequent unknown tokens, $m(w_m) = \epsilon$, are determined and added to an auxiliary lexicon $\mathcal{L}^+$. For all tokens w_m of the auxiliary lexicon the possible tags are manually assigned and denoted by $\mathcal{T}_{w_m}$. If there is more than one tag possible, an adequate tag distribution needs to be assigned as well. Therefore, two approaches are utilized.

First, if at least one word $\hat{w}_k$ of the manually annotated training corpus has the same POS tag set as the manually assigned set $\mathcal{T}_{w_m}$, the cumulated tag distribution of those words is taken. Hence, the lexical probability is refined as

$$\hat{P}_{\mathcal{L}^+}(t \mid w_m) = \frac{|\{k \mid \hat{t}_k = t \wedge \mathcal{T}_{\hat{w}_k} = \mathcal{T}_{w_m})\}|}{|\{k \mid \mathcal{T}_{\hat{w}_k} = \mathcal{T}_{w_m}\}|},$$

where $\mathcal{T}_{\hat{w}_k} = \{\hat{t}_l \mid \hat{w}_l = \hat{w}_k\}$.

Second, if there is no word with the same set of possible tags in the training text, further manual annotation is performed. A reliable amount of such tokens is manually annotated in the large social media corpus SL. The resulting tag distribution is assigned to such unknown tokens.

Such unknown tokens are often shortened verbs or wrongly uncapitalized tokens. The following example illustrates that particular case. Consider the word *benutz* (*use*), which in a standardized text is just used as an imperative content word (VVIMP). In social media texts the word is also used as short form for the verb inflections *benutze* (*[I] use*) and *benutzt* (*[he] uses*). Hence, the resulting tag distribution from manual annotation of the large social media corpus results in $P(VVFIN) = 59\%, P(VVIMP) = 32\%$ and $P(VVPP) = 9\%$.

IV. Results

Different criteria are analyzed to evaluate the proposed approach. First, WebTagger is compared to four state-of-the-art POS taggers, considering German Web comments. Second, the performance improvements for each proposed enhancement step are demonstrated. Furthermore, the improvement by using manually annotated Web comments for training is pointed out. Particularly, we show that using non-standardized texts for training does not lead to a significant degradation when applying WebTagger to standardized newspaper texts. Finally, we study the transferability to different social media text types, where the taggers are not trained on the particular type.

A. Corpora

Two corpora are used for the purpose of training our new social media corpus *WebTrain* and a newspaper text corpus. *WebTrain* corpus contains 429 Web comments collected from *Heise.de*, which is a popular German newsticker site treating different technological topics. Each of 36,000 token is annotated with manually validated POS tags and lemmas. Annotation rules, particularly for social media text characteristics, are taken from [16]. The average POS tag ambiguity of tokens contained in the corpus is 2. This is significantly higher as the ambiguity in German newspaper texts, e.g., 1 for the *TIGER* corpus containing 890,000 tokens. In order to provide a sufficiently large training data amount, we combine *WebTrain* with the *TIGER* treebank [19] newspaper text corpus. We call that joint-domain training. *WebTrain* texts contain 18% trigrams, that never occur in the newspaper corpus *TIGER*. Those trigrams constitute 6% frequency of all *WebTrain* trigram counts. Both results motivate the need of text genre specific training data for reliable estimation of transition probabilities, e.g., for trigrams.

To have a deeper look in the general applicability of *WebTagger* for social media texts, an additional corpus *WebTypes* is used. It is composed of roughly 4,000 tokens, where comments from different Web sites and a corpus extract from the *Dortmunder chat corpus BalaCK 1-b* [20] are manually annotated. Three different types of social media texts are represented, YouTube comments, blog comments, and chat messages. Further corpus statistics can be found in [15].

B. Cross validation

A 10-fold cross validation is performed to evaluate the tagging accuracy of our approach, compared to state-of-the-art taggers. Therefore, *WebTrain* is divided into ten equally sized subsets which are created by randomly selected sentences. WebTagger and three state-of-the art taggers are trained on a combination of nine subsets and *TIGER* data in each validation step. The remaining subset is used for testing. The selected taggers are TreeTagger [4], TnT [5], and Stanford [1]. In a previous study [15] we evaluated the performance of the mentioned taggers for social media texts in more detail. Total tagging accuracies and accuracy rates achieved for known tokens and unknown tokens are determined. Mean accuracies and their standard deviation are given in Table II. WebTagger significantly exceeds the mean tagging accuracy compared to all state-of-the-art taggers. During the ten test runs we

TABLE II
RESULTS FOR 10-FOLD CROSS VALIDATION TRAINED ON JOINT-DOMAIN DATA USING *WebTrain*.

	WebTagger	TreeTagger	TnT	Stanford
Total	**94.09 ± 0.37**	93.72 ± 0.49	93.63 ± 0.37	93.18 ± 0.32
Known	**95.15 ± 0.37**	95.83 ± 0.43	95.81 ± 0.51	95.61 ± 0.40
Unknown	**72.75 ± 2.25**	67.98 ± 3.14	70.58 ± 2.08	68.14 ± 1.97
Percentage unknowns	**4.72 ± 0.40**	7.58 ± 0.75	8.65 ± 0.62	8.81 ± 0.62

perform 30 single comparisons with WebTagger. WebTagger performs in 28 of 30 cases better. Particularly, the accuracy on unknown tokens can be improved by our approach. Note that, the tagging accuracy for known tokens is slightly worse compared to state-of-the art taggers. But at the same time the number of unknown tokens is significantly reduced from 2.9 up to 4.1 percentage points compared to the state-of-the art taggers. Moreover, the accuracy for unknown tokens is increased. The main contribution for this is given by our approach combining prefix and suffix lexical probabilities by the arithmetic mean, which makes the method more robust for unknown tokens. Overall, considering the noisy characteristics of social media texts, a considerable enhancement is achieved with WebTagger. Precision and recall rates for each tag class are determined to investigate the tag/class-specific accuracies. Applying WebTagger leads to a mean precision of 0.86 with a standard deviation of 0.2. On average, a recall of 0.84 with standard deviation 0.23 is achieved. Considering the equally weighted combination of both measures, our approach results in a mean f_1-measure of 0.84.

The ten most frequently confused tag pairs for our approach are further investigated. The results are depicted in Table III.

TABLE III
MOST FREQUENTLY CONFUSED TAG CLASSES.

Correct	**Predicted**	**Frequency**
NE	**NN**	147
NN	**NE**	102
VVFIN	**VVINF**	90
KOM	ADV	58
FM	**NE**	58
NN	ADJA	46
PDS	ART	42
VVFIN	**VVPP**	40
VVINF	VVFIN	38
PRELS	ART	38

Bold classes also occur in the top ten confused tag classes, evaluated only for unknown tokens. Tagging errors are represented by absolute values/frequencies and calculated over all testing sets with approximately 36,000 tokens. The top two confusion pairs *noun* (NN) and *named entity* (NE) account for 12% of the errors. This is not a particular effect for social media texts, since it also occurs for newspaper texts. To distinguish proper nouns from named entities is done by named entity recognition and can not be solved by general POS taggers. Interchanging a *finite verb* (VVFIN) and a *non-finite verb* (VVINF) is caused by a non-local dependency particularly in German. This is also reported for state-of-the-art taggers and illustrated in [4]. Noticeable is the occurrence of tag confusion between *foreign language* (FM) and *named entity* (NE).

Social media texts are often multilingual and contain text parts written in different language, e.g., a German Web comment contains English text segments (FM). The tokens of such text segments are annotated as *foreign language* (FM). Due to the missing prefix/suffix information of such tokens, this leads to tagging errors. Frequent tag confusion between *noun* (NN) and *attributive adjective* (ADJA) results from missing noun capitalization, which causes a valid adjective, from self created tokens or token transformations.

Furthermore, we investigate the influence of training data selection and parameter estimation adaptions for lexical probabilities. Results are depicted in Table IV.

TABLE IV
INFLUENCE OF TRAINING DATA AND ESTIMATION METHODS

	Method	**Accuracy** (%)
Training	*TIGER* newspaper corpus	87.59 ± 1.21
	TIGER + *WebTrain* corpus, i.e., (+)	93.38 ± 0.42
Estimation	(+) + normalization ($t(w)$)	93.61 ± 0.44
	(+) + normalization ($t(w)$) + word classes ($\mathcal{W}_r$), i.e., ($\star$)	93.91 ± 0.39
	($\star$) + auxiliary lexicon ($\mathcal{L}^+$)	**94.09 ± 0.37**

A significant improvement is achieved by adding text genre specific training data, i.e., Web comments. We discuss this effect in detail in Section IV-C. The introduction of text normalization and regular expressions to build word classes leads to a significant improvement of 0.57 percentage points. Additional usage of an auxiliary lexicon, further increases accuracy about 0.18 percentage points.

All depicted results use a combined prefix/suffix lexicon to estimate lexical probabilities for (still) unknown tokens. Previous studies have shown, that adding prefix information for automatic tagging of newspaper texts only leads to little improvement of 0.05 percentage points, see [4]. In our approach running the tagger only with a suffix lexicon results in 0.3 and only with a prefix lexicon in 0.7 percentage points performance loss.

C. Influence of text genre-specific training data

To further investigate the influence of using text genre specific data, i.e., Web comments for training, we train our model based on different training corpora. First, we train our

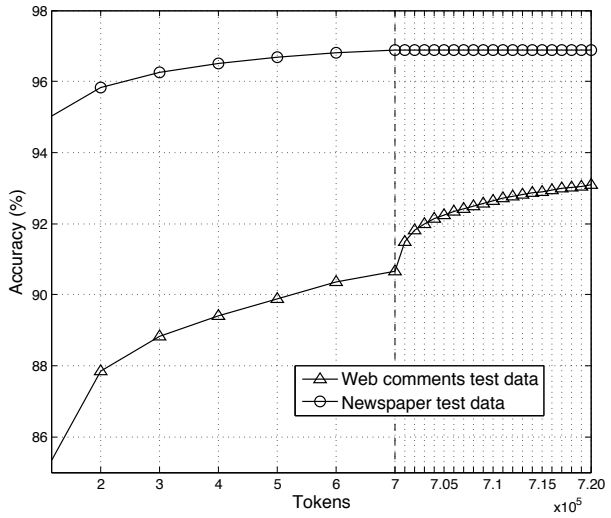

Fig. 1. Influence of additional newspaper/Web comment training, tested on Web comment and newspaper texts.

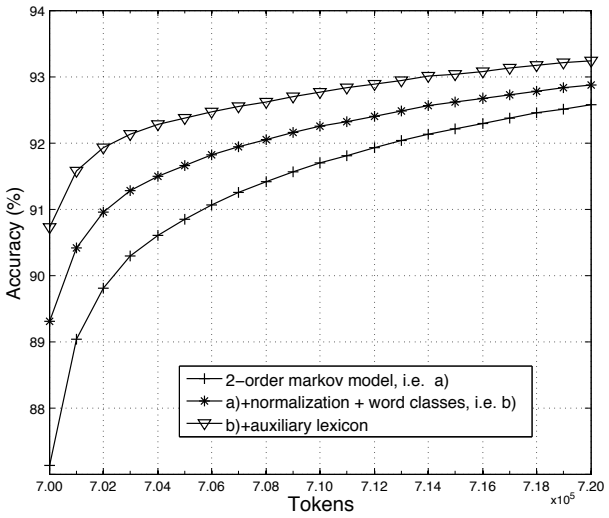

Fig. 2. Stepwise parameter estimation adaptions for increasing Web comment training data.

model exclusively on newspaper texts. We stepwise increase the amount of training data from 100,000 to 700,000 tokens. In each step we randomly choose sentences comprising 100,000 tokens. This is performed 100 times and data is added to the data selected in the previous step. Hence, the model is trained on 100 different samples in each step. Second, in twenty steps 1,000 up to 20,000 Web comment tokens are combined with a sample set of 700,000 newspaper training tokens. Here, we choose the newspaper training sample (700,000 tokens) achieving mean tagging accuracy, when tested on Web comments. Additional Web comment tokens are chosen randomly, sentence wise from *WebTrain*. Again we select 100 sample sets, in the same way as for the newspaper training and train our model on such data for each iteration step. Testing is performed on the remaining data, a fixed test set of Web comments with approximately 6,000 tokens. Mean results over 100 different trainings per point are depicted in the curve marked with $\triangle$ in Figure 1. The plot contains different x-axis scalings for the left and right area next to the black vertical line to better illustrate the results. Significant slope increase can be observed in this point, which proves the success by using text genre specific training data for the task of POS tagging for Web comments. Using 20,000 Web comment tokens results in approximately 2.4 percentage points performance improvement on average. Hence, little effort of manual annotation leads to a significant performance improvement. Increase of 600,000 newspaper training tokens results in approximately 5.8 percentage points improvement solely.

Furthermore, we show that including grammatically non-standardized texts as training data does not negatively effect the annotation of standardized text by means of the proposed approach. Random sentences are chosen from the newspaper *TIGER* corpus to create a test set of 90,000 newspaper tokens. We use WebTagger trained on the different training corpora to tag the newspaper data. The curve marked with $\circ$ in Figure 1 illustrates the results. Results proof that adding 20,000 Web comment tokens for training do not effect tagging accuracy for standardized texts essentially.

Comparison of tagging accuracies for Web comments and newspaper texts states that the tagging accuracy on standardized text can not be achieved when applying our approach to Web comments. However, the performance difference can be reduced from approximately 10 percentage points to 4 percentage points by increasing the amount of training data from 100,000 tokens to 720,000 tokens in total. Furthermore, matching the slope of both curves for the left area, states that increasing the amount of newspaper training data is more substantial for the application to Web comments compared to the application to newspaper texts. Tagging accuracy can be improved by 2 percentage points for newspaper data and 5.8 percentage points tested on Web comments by adding the same amount of newspaper training data.

Training our model on 700,000 *TIGER* tokens leads to similar results, when tested on newspaper data compared to TreeTagger results reported in [10]. For 90,000 randomly selected testing sentences chosen from the *TIGER* corpus, WebTagger achieves 96.9% accuracy on average.

Finally, the interaction between the amount of training data and the different adaption method for lexical probability estimation is illustrated in Figure 2. For testing the same 6,000 test tokens like in Figure 1 are used. We stepwise adapt the lexical parameter estimation method by our proposed methods, similarly to the procedure performed in Table IV. Significant impact of introducing text normalization and word classes is observed over the whole training data range. Using an auxiliary lexicon leads to a significant performance increase,

TABLE V
TAGGER EVALUATION FOR DIFFERENT TEXT TYPES TRAINED ON JOINT-DOMAIN DATA.

	#Tokens	WebTagger	TreeTagger	TnT	Stanford
WebTrain test	3,628	**94.09 ± 0.37**	93.72 ± 0.49	93.63 ± 0.37	93.18 ± 0.32
Chat messages	1,728	**91.75 ± 0.09**	89.12 ± 0.18	87.96 ± 0.11	87.81 ± 0.16
YouTube comments	1,463	**86.90 ± 0.19**	84.03 ± 0.24	81.18 ± 0.19	81.23 ± 0.16
Blog comments	815	**93.56 ± 0.29**	91.35 ± 0.18	90.46 ± 0.12	90.29 ± 0.17

particularly for a small amount of training data. Comparing the slopes of the curves marked with ∇ and $\star$ illustrates that the sufficient training data amount is much higher to compensate the improvement achieved by normalization and word classes methods. In total, all estimation adaptions can be partially compensated by adding additional Web comment training data at least for this test sample. This has to be studied in more detail for different test samples. However, manual annotation of complete texts for fully supervised training is a very time consuming step. Creating an auxiliary lexicon with our proposed method shows a better trade-off between time for annotation and improvement in tagging accuracy.

D. Transfer to other social media text types

In this subsection, we study the application of the proposed WebTagger to different social media text types, where the tagger is not trained on the particular type. To illustrate the improvements, Table V shows tagging accuracies and standard deviations for WebTagger and the three selected state-of-the art taggers. All taggers are trained on the joint-domain cross validation data described before. We compare the results for the particular Web comment test data to results achieved for blog comments, chat messages and YouTube comments from *WebTypes* corpus, introduced in Subsection IV-A.

Application of WebTagger leads to a consistent performance increase between approximately 2 and 6 percentage points for different social media text types. Best improvements can be observed for YouTube comments, which are highly characterized by a dialogue form and social media text characteristics, such as emoticons, word shortenings or letter iterations. Even though considerable improvement is achieved, the tagging accuracy of 86.9% is the lowest compared to all other types, due to the low text standardization. Overall, WebTagger outperforms the state-of-the art taggers for all social media text types.

Figure 3 shows the influence of additional Web comment training data to the different social media text types. The accuracy improvements over the different training data amounts are depicted in the corresponding curves. For all social media types the stepwise addition of *WebTrain* training data leads to a consistent accuracy increase. For *WebTypes* related text types, which show more social media text characteristics, the slope of the curves is higher compared to the particular training data type *WebTrain* (test, 6,000 tokens). Increasing the amount of Web comment training data leads to a significant performance increase, particularly for

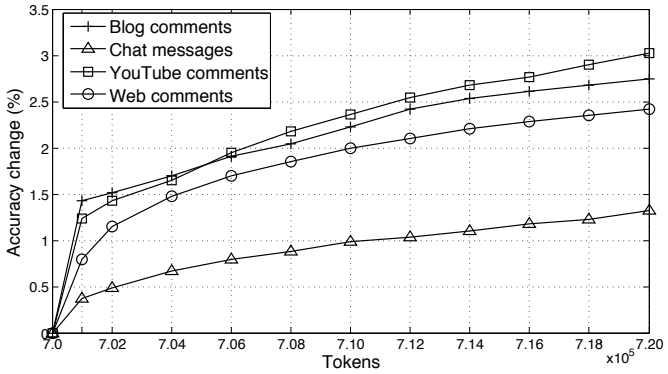

Fig. 3. Influence of additional Web comment training for different social media texts.

blog comments and YouTube comments. Results approve that general social media text characteristics can be learned from Web comments (*Heise*). In summary, the results from Table V and Figure 3 show that the adapted parameter estimation methods combined with a sufficient amount of Web comment training data leads to adequate tagging accuracies for social media texts in general. Results clearly demonstrate that the proposed tagger can successfully be applied to other texts belonging to the social media text genre.

Note that we exclude twitter messages from this scope, since this subset can not be addressed suitably with the presently developed method. Due to their special characteristic given by hard distractions to 140 characters, the proposed method needs to be further adapted.

E. Transfer to other languages

The basic model and parameter estimation enhancements of the proposed WebTagger are language independent. It is adapted to the social media text characteristics in general, e.g., emoticons or character iterations. However, considering all minor effects that depend on language specific properties requires some additional effort, e.g., for adapting the normalization function. Moreover, language specific training would require an additional supervised social media text corpus. For the corpus annotation all described annotation rules can be used analogously. Evidently, POS tags need to be mapped to the language specific tag set.

V. Conclusion

A new POS tagger, WebTagger, designed for the annotation of social media texts has been presented. It yields a minimum improvement of 2.2 percentage points for different social media text types compared to state-of-the art taggers. Furthermore, WebTagger performs the best with an average accuracy of 94% evaluated in a cross validation on German Web comments. Our approach basically differs from other statistical Markov model taggers in estimation of lexical probabilities for unknown tokens. Before word classes realized by regular expressions and a prefix and suffix lexicon is adequately combined, a word preprocessing for text normalization is performed. Additionally, the usage of a semi-supervised auxiliary lexicon is proposed. Altogether, lexical probability distributions are estimated more accurately for social media texts.

Furthermore, the influence of manually annotated text genre specific training data, i.e., social media texts, is investigated. Considerable improvement is achieved by using only a small amount of 20,000 tokens as additional data for supervised training. Using such training data enables for reliable transition probability estimates by learning the different grammatical structure of social media texts.

In our approach we exemplarily use German social media texts. WebTaggers basic model and parameter estimation enhancements are language independent. However, we recommend a language specific training which requires an additional supervised social media text corpus.

Acknowledgment

This work was partially supported by the Project House HumTec at RWTH Aachen University, Germany. We would like to thank Phillip Vaßen for his contribution.

References

[1] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich Part-of-Speech Tagging With a Cyclic Dependency Network," in *Proceedings of Human Language Technology Conference*, 2003, pp. 173–180.

[2] P. Gadde, L. V. Subramaniam, and T. A. Faruquie, "Adapting a WSJ Trained Part-of-Speech Tagger to Noisy Text: Preliminary Results," in *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, 2011, pp. 5:1–5:8.

[3] H. Schmid, "Probabilistic Part-of-Speech Tagging Using Decision Trees," in *Proceedings of International Conference on New Methods in Language Processing*, 1994, pp. 44–49.

[4] ——, "Improvements in Part-of-Speech Tagging With an Application to German," in *Proceedings of the ACL SIGDAT-Workshop*, 1995, pp. 47–50.

[5] T. Brants, "TnT – A Statistical Part-of-Speech Tagger," in *Proceedings of the 6th Applied Natural Language Processing Conference*, 2000, pp. 224–231.

[6] A. Schiller, S. Teufel, C. Stöckert, and C. Thielen, "Guidelines für das Tagging deutscher Textcorpora mit STTS," 1999, university of Stuttgart.

[7] J. Giménez and L. Màrquez, "Svmtool: A General POS Tagger Generator Based on Support Vector Machines," in *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004, pp. 43–46.

[8] H. Schmid, "Part-of-Speech Tagging With Neural Networks," in *Proceedings of the 15th Conference on Computational Linguistics*, 1994, pp. 172–176.

[9] M. Volk and G. Schneider, "Comparing a statistical and a rule-based tagger for German," in *Proceedings of the 4th Conference on Natural Language Processing*, 1998, pp. 125–137.

[10] E. Giesbrecht and S. Evert, "Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus," in *Proceedings of the Fifth Web as Corpus Workshop*, 2009, pp. 27–35.

[11] A. Mikheev, "Automatic Rule Induction for Unknown Word Guessing," *Computational Linguistics*, vol. 23, pp. 405–423, 1997.

[12] H. Schütze, "Distributional Part-of-Speech Tagging," in *Proceedings of 7th Conference of the European Chapter of the Association for Computational Linguistics*, 1995, pp. 141–148.

[13] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, and N. Schneider, "Part-of-Speech Tagging for Twitter: Word Clusters and Other Advances," School of Computer Science, Carnegie Mellon University, Tech. Rep., 2012.

[14] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, "Part-of-Speech tagging for Twitter: annotation, features, and experiments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 2011, pp. 42–47.

[15] M. Neunerdt, M. Reyer, and R. Mathar, "Part-of-Speech Tagging for Social Media Texts," in *Proceedings of The International Conference of the German Society for Computational Linguistics and Language Technology*, 2013.

[16] B. Trevisan, M. Neunerdt, and E.-M. Jakobs, "A Multi-level Annotation Model for Fine-grained Opinion Detection in German Blog Comments," in *Proceedings of KONVENS 2012*, 2012, pp. 179–188.

[17] M. Beißwenger, M. Ermakova, A. Geyken, L. Lemnitzer, and A. Storrer, "A TEI Schema for the Representation of Computer-mediated Communication," *Journal of the Text Encoding Initiative*, no. 3, pp. 1–31, 2012. [Online]. Available: http://jtei.revues.org/476

[18] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, pp. 81–106, 1986.

[19] S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, and H. Uszkoreit, "TIGER: Linguistic Interpretation of a German Corpus," *Research on Language & Computation*, pp. 597–620, 2004.

[20] M. Beißwenger, "Corpora zur computervermittelten (internetbasierten) Kommunikation," *Zeitschrift für germanistische Linguistik*, vol. 35, pp. 496–503, 2007.

N-gramas sintácticos no-continuos

Grigori Sidorov

***Resumen*—En este artículo presentamos el concepto de los n-gramas sintácticos no-continuos. En nuestros trabajos previos hemos introducido un concepto general de los n-gramas sintácticos, es decir, los n-gramas que se construyen siguiendo las rutas en un árbol sintáctico. Su gran ventaja consiste en que permiten introducir información puramente lingüística (sintáctica) en los métodos computacionales de aprendizaje automático. Su desventaja está relacionada con la necesidad de realizar el análisis sintáctico automático previo. También hemos demostrado que la aplicación de los n-gramas sintácticos en la tarea de atribución de autoría da mejores resultados que el uso de los n-gramas tradicionales. Sin embargo, en dichos trabajos sólo hemos considerado los n-gramas sintácticos continuos, es decir, durante su construcción no se permiten bifurcaciones en las rutas sintácticas. En este artículo, estamos proponiendo a quitar esta limitación, y de esa manera considerar todos los sub-árboles de longitud *n* de un árbol sintáctico como los n-gramas sintácticos no-continuos. Cabe mencionar que los n-gramas sintácticos continuos son un caso particular de los n-gramas sintácticos no-continuos. El trabajo futuro debe mostrar qué tipo de n-gramas es más útil y para qué tareas de PLN. Se propone la manera formal de escribir un n-grama sintáctico no-continuo usando paréntesis y comas, por ejemplo, "*a b [c [d, e], f]*". También presentamos en este artículo ejemplos de construcción de n-gramas sintácticos no-continuos para los árboles sintácticos obtenidos usando FreeLing y el *parser* de Stanford.**

***Palabras clave*—Modelo de espacio vectorial, n-gramas, n-gramas sintácticos continuos, n-gramas sintácticos no-continuos.**

Non-continuous Syntactic N-grams

***Abstract*—In this paper, we present the concept of non-continuous syntactic n-grams. In our previous works we introduced the general concept of syntactic n-grams, i.e., n-grams that are constructed by following paths in syntactic trees. Their great advantage is that they allow introducing of the merely linguistic (syntactic) information into machine learning methods. Certain disadvantage is that previous parsing is required. We also proved that their application in the authorship attribution task gives better results than using traditional n-grams. Still, in those works we considered only continuous syntactic n-grams, i.e., the paths in syntactic trees are not allowed to have bifurcations. In this paper, we propose to remove this limitation, so we consider all sub-trees of length *n* of a syntactic tree as non-continuous syntactic n-grams. Note that continuous syntactic n-grams are the particular case of non-continuous syntactic n-grams. Further research should show which n-grams are more useful and in which NLP tasks. We also propose a formal manner of writing down (representing) non-continuous syntactic n-grams using parenthesis and commas, for example, "*a b [c [d, e], f]*". In this paper, we also present examples of construction of non-continuous syntactic n-grams on the basis of the syntactic tree of the FreeLing and the Stanford parser.**

***Index Terms*—Vector space model, n-grams, continuous syntactic n-grams, non-continuous syntactic n-grams.**

Manuscrito recibido 12 de junio de 2013. Manuscrito aceptado para su publicación 23 de septiembre de 2013.

Grigori Sidorov trabaja en el Centro de Investigación en Computación (CIC), Instituto Politécnico Nacional (IPN), Av. Juan de Dios Bátiz, s/n, esq. Othón de Mendizábal, Zacatenco, 07738, México DF, México (e-mail: www.cic.ipn.mx/~sidorov).

I. Introducción

En el análisis automático de lenguaje natural (procesamiento de lenguaje natural, PLN) y en la lingüística computacional [1] cada vez son más populares los métodos relacionados con el aprendizaje automático computacional (*machine learning*, en inglés). Aplicando estos métodos se obtienen resultados cada vez mejores [2], [3].

La intención principal detrás de la aplicación de los métodos de aprendizaje automático es tratar de modelar la formulación de hipótesis por parte de los lingüistas y su posterior verificación. En este caso se sustituye la intuición humana por las grandes cantidades de datos textuales, posiblemente con marcas adicionales hechas manualmente, y por métodos sofisticados de aprendizaje, que se basan en las matemáticas y en las estadísticas. De ninguna manera se pretende sustituir a los lingüistas en el proceso de investigación, sino desarrollar herramientas que puedan ser útiles para ellos. Por otro lado, la aplicación de métodos de aprendizaje automático permite la evaluación exacta de las hipótesis, la reproducción de los resultados y hasta cierto punto convierte a la lingüística computacional en una ciencia más exacta. En este sentido, algunas partes de la lingüística computacional ya requieren procedimientos empíricos, cuando la hipótesis formulada se verifica utilizando los experimentos computacionales basados en datos, y no sólo en la intuición de hablantes nativos o del propio experimentador.

Los métodos más utilizados en la lingüística computacional son en su mayoría métodos supervisados, es decir, se utilizan datos marcados manualmente para realizar entrenamiento. Otra posibilidad está relacionada con los métodos no supervisados, cuando el propio sistema debe aprender directamente de los datos. Claro está que es mucho más difícil utilizar los métodos no supervisados, porque en este caso en ningún lado se encuentra una intervención humana; normalmente para poder aplicar estos métodos se requiere una gran cantidad de datos.

Por lo mencionado anteriormente, en la lingüística computacional ya son aplicables los conceptos de precisión y especificidad (*precision* y *recall*, en inglés), que miden la viabilidad de una hipótesis de manera formal. Igual que el concepto de línea base (*baseline*), que corresponde al método

comúnmente aceptado del estado del arte que resuelve el mismo problema, y que debe ser superado por la hipótesis propuesta. Normalmente la línea base es un método no muy sofisticado. Dado que estamos hablando de datos marcados por seres humanos de manera manual y que se usan para la verificación del método, se utiliza el concepto de estándar de oro (*gold standard*). Como es marcado de manera manual, se supone que no tiene errores (de aquí viene la idea que es de "oro"), y si el sistema lo puede alcanzar, entonces el sistema funciona realmente muy bien. Una cuestión interesante está relacionada con la concordancia entre los juicios humanos, es decir, si los propios seres humanos marcan de manera diferente algún fenómeno lingüístico, entonces no podemos esperar que la máquina resuelva correctamente este mismo fenómeno. Aquí viene el concepto de la línea tope (*top line*, en inglés). Por lo mismo, es recomendable usar varios evaluadores, y no sólo uno, para tener juicios mejor fundamentados y menos sesgados.

Para realizar los experimentos se utiliza comúnmente la técnica llamada validación cruzada de *k* volúmenes (k-*fold cross validation*), donde *k* tiene un valor numérico, normalmente igual a 10. La técnica consiste en dividir todos los datos en 10 (o *k*) volúmenes. Primero se utiliza el volumen 1 para la evaluación de los resultados del sistema, y los volúmenes 2 a 10 para el entrenamiento, después se elige el volumen 2 para la evaluación, y los otros nueve volúmenes para el entrenamiento, y así sucesivamente. De esta manera el sistema es evaluado 10 veces sobre datos diferentes y se entrena 10 veces sobre datos algo diferentes, finalmente se toma el promedio de las 10 evaluaciones. Nótese que es muy importante no realizar la evaluación sobre los mismos datos, sobre los cuales se hizo el entrenamiento: por eso es la división en *k* volúmenes.

Para realizar todos los procedimientos descritos, tenemos que representar el problema de manera formal. El modo más utilizado de representación es el modelo de espacio vectorial [2], [4]. Se representa el problema como un problema de clasificación automática en un espacio, que es precisamente el espacio vectorial. Los objetos se representan como conjuntos de valores de características, es decir, a cada objeto le corresponde un vector de dichos valores (de aquí el término "espacio vectorial"). Eso significa que cada objeto es un punto en el espacio multidimensional de características. Es muy fácil imaginar este modelo para el caso de dos características (dos dimensiones). Para un mayor número de dimensiones simplemente hay que suponer que es similar. Después de definir el espacio, se define una métrica en este espacio. Normalmente es la métrica de similitud de objetos, definida por la similitud de coseno [2], [4]. La idea de dicha similitud es: más se parecen los dos objetos entre sí, menor es el ángulo entre ellos en el espacio definido, y por lo tanto mayor es el coseno de este ángulo.

Ahora bien, la siguiente pregunta es: cómo elegir las características para definir el espacio vectorial. En este momento empiezan a prevalecer las consideraciones lingüísticas: es precisamente la parte lingüística que determina las características que podemos elegir. Por ejemplo, la idea más sencilla utilizada en recuperación de información es utilizar todas las palabras en varios documentos como sus características, y después comparar esos documentos: cuantas más palabras "iguales" tienen dos documentos, más se parecen entre sí estos documentos. Así se construye el espacio vectorial para la tarea de recuperación de información.

Obviamente en este caso, las ideas de qué tipo de información lingüística se puede utilizar se restringen por el requisito formal de utilizar el modelo de espacio vectorial, es decir, la obligación de representar los objetos como los conjuntos de valores de características.

La siguiente posibilidad más utilizada en la práctica, y que ya tiene cierto fundamento lingüístico, es la idea de utilizar n-gramas como características en el modelo de espacio vectorial. El concepto de n-grama es realmente muy sencillo: son las secuencias de palabras (u otros tipos de elementos) según aparecen en los textos. Entonces, el fundamento es que se tome en cuenta la información sintagmática de las palabras.

Sin embargo, sería muy provechoso utilizar conocimiento aún más "lingüístico", es decir, que involucre más información propiamente lingüística, por ejemplo, como en [5], [6], [7]. Como un camino en esta dirección, en nuestros trabajos anteriores [4], [8], [9], [10] hemos propuesto un nuevo concepto de n-gramas, que contiene aún una mayor información de naturaleza lingüística que los n-gramas tradicionales: n-gramas sintácticos. La idea de los n-gramas sintácticos consiste en construirlos siguiendo la ruta en el árbol sintáctico. De esa manera los n-gramas sintácticos siguen siendo n-gramas, pero permiten introducir información sintáctica [11] en los métodos de aprendizaje automático.

Sin embargo, en todos esos trabajos hemos propuesto obtener un n-grama sintáctico como un fragmento de una ruta continua, no se permiten bifurcaciones en la ruta —más adelante presentaremos ejemplos de eso. En este artículo vamos a presentar el concepto de n-gramas sintácticos no-continuos, es decir, siguiendo la ruta en el árbol sintáctico, se permite entrar en las bifurcaciones y regresar.

La estructura del resto del artículo es la siguiente. Primero discutimos el concepto de los n-gramas sintácticos continuos, tal como se presentó en nuestros trabajos anteriores. Después presentamos el concepto de los n-gramas sintácticos no-continuos. En las siguientes secciones vamos a considerar los ejemplos de extracción de los n-gramas sintácticos continuos y no-continuos para el español y el inglés. Finalmente, presentaremos las conclusiones.

II. N-GRAMAS SINTÁCTICOS CONTINUOS

En nuestros trabajos anteriores [4], [8], [9], [10] hemos introducido el nuevo concepto de los n-gramas sintácticos, es decir, n-gramas obtenidos siguiendo las rutas en un árbol

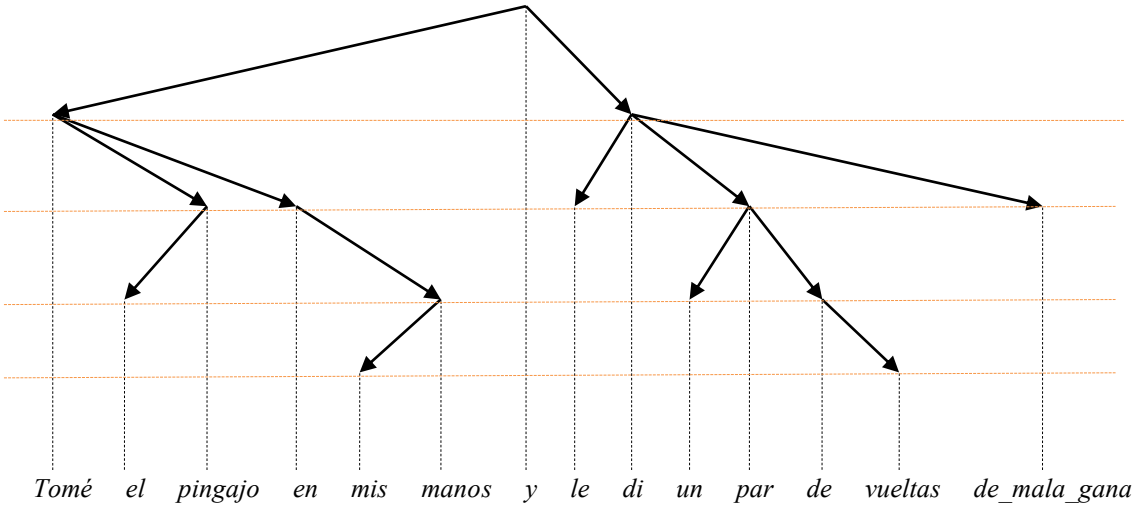

Fig. 1. Árbol sintáctico (analizado por FreeLing) representado con dependencias

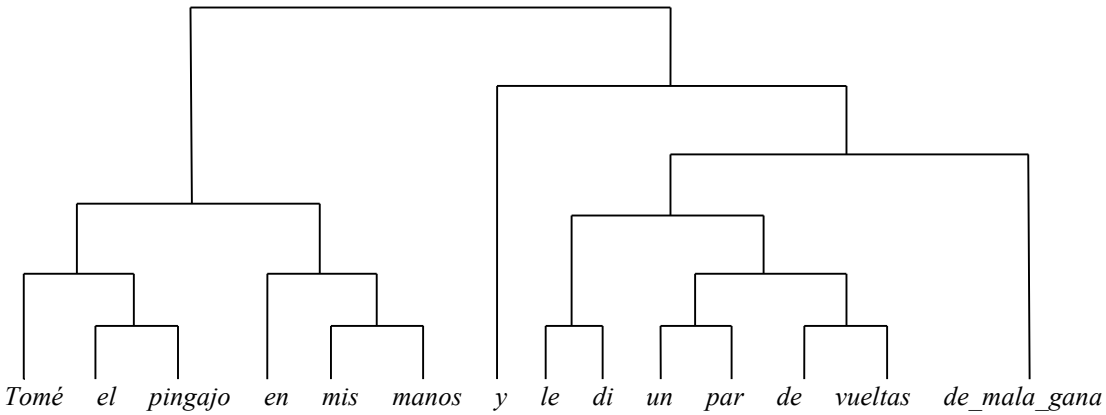

Fig. 2. Árbol sintáctico (analizado por FreeLing) representado con constituyentes

sintáctico. Un árbol sintáctico de una frase muestra[1] se presenta en las fig. 1 y 2, utilizando el formalismo de dependencias y constituyentes. Cabe mencionar que la idea de utilizar información estructural de relaciones de palabras en tareas específicas se ha presentado anteriormente [12], [13], [14], [15], [16], sin embargo, en ninguno de estos trabajos se ha generalizado, ni se ha relacionado con la idea de los n-gramas.

El trabajo [17] ha propuesto una idea similar en el campo de análisis semántico, donde la utilidad de la información sintagmática se demuestra en tareas muy específicas de: 1) "*semantic priming*", es decir, los experimentos psicolingüísticos sobre la similitud de palabras, 2) detección de sinónimos en las pruebas TOEFL, y 3) ordenamiento de sentidos de palabras según su importancia.

En nuestra opinión, este trabajo no tuvo mucha resonancia en otras tareas de PLN precisamente por no relacionar la información sintáctica con los n-gramas, que es la herramienta principal en la gran mayoría de tareas, ni por mostrar su utilidad en otras tareas que no son tan específicamente orientadas a la semántica.

Anteriormente, hemos propuesto varios tipos de n-gramas sintácticos con base en qué elementos los constituyen:

- Elementos léxicos (palabras, lemas, o raíces),
- Etiquetas de las categorías gramaticales (*POS tags*),
- Nombres de las relaciones sintácticas (*SR tags*),
- Caracteres,
- N-gramas sintácticos mixtos (combinaciones de los tipos anteriores).

Cabe mencionar que para la construcción de los n-gramas sintácticos de caracteres es necesario primero construir los n-gramas sintácticos de palabras y después basándose en éstos construir los n-gramas de caracteres. Es cuestión de trabajo futuro verificar si los n-gramas sintácticos de caracteres son útiles para algunas tareas de PLN. Resulta que la aplicación de

[1]Frase de una de las obras de A. Conan-Doyle.

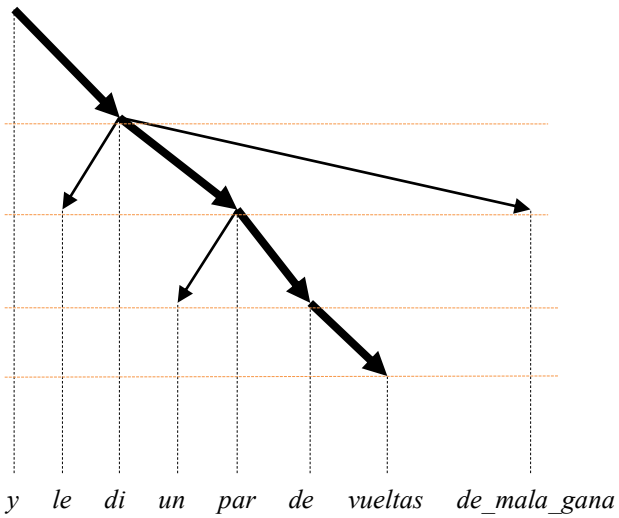

Fig. 3. N-gramas sintácticos continuos en el fragmento del árbol sintáctico: ejemplo de un 5-grama

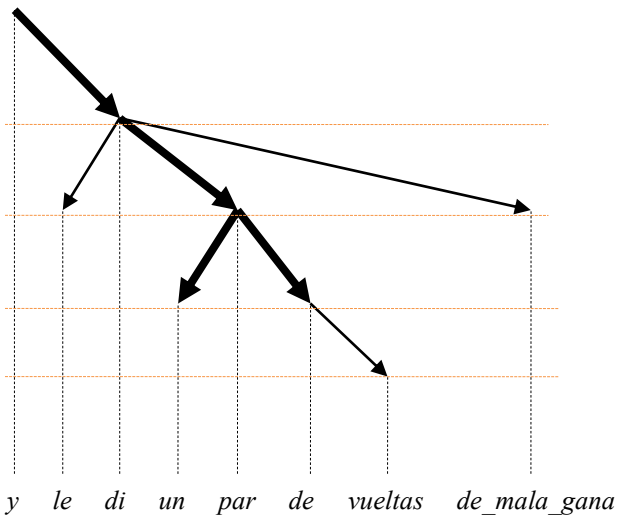

Fig. 4. N-gramas sintácticos no-continuos en el fragmento del árbol sintáctico: ejemplo de un 5-grama

los n-gramas tradicionales de caracteres presenta buenos resultados en algunas tareas, por ejemplo, en la tarea de detección de autoría [17]. Sin embargo, desde nuestro punto de vista, la aplicación de los n-gramas de caracteres es hasta cierto punto anti-intuitivo, y falta analizar las razones de su funcionamiento aceptable.

Respecto a los n-gramas mixtos, se deberá analizar a futuro qué combinaciones de los elementos: palabras, *POS-tags*, *SR-tags*, y en qué posiciones: al inicio, en medio de un n-grama, o al final, está dando mejores resultados.

En [17] se menciona la idea de ponderar las relaciones entre los elementos de un n-grama sintáctico. Esa idea no nos parece aplicable directamente en el contexto de modelos de espacio vectorial, donde los n-gramas son las características (dimensiones del espacio). Sin embargo, esa idea puede ser útil en el momento de calcular los pesos de n-gramas sintácticos, aparte de los valores tradicionales de *tf-idf*.

En nuestros trabajos anteriores [18] hemos demostrado que los n-gramas sintácticos pueden dar mejores resultados que los n-gramas tradicionales. Lo analizamos para el problema de detección de autoría. Sin embargo, los n-gramas sintácticos pueden ser utilizados en cualquier tipo de problemas donde se utilizan los n-gramas tradicionales, porque igual permiten la construcción del modelo de espacio vectorial. La desventaja de los n-gramas sintácticos consiste en el hecho que se requiere el análisis sintáctico automático previo, lo que toma cierto tiempo de procesamiento, aunque no es una limitación muy seria. También no para todos los idiomas existen analizadores sintácticos automáticos, pero sí para los idiomas con mayor presencia en el mundo, como el español o el inglés.

Independientemente del tipo de elementos que constituyen los n-gramas sintácticos, todos los n-gramas sintácticos considerados en nuestros trabajos anteriores, son continuos. Eso quiere decir que la ruta sintáctica que estamos siguiendo nunca tiene bifurcaciones, véase la comparación de la fig. 3 y

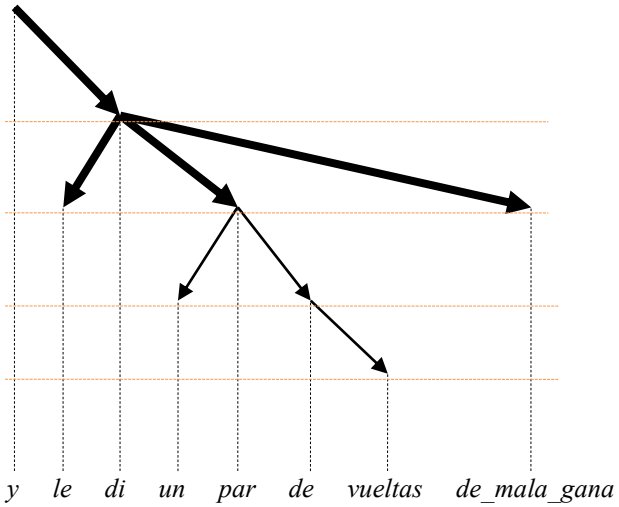

Fig. 5. N-gramas sintácticos no-continuos en el fragmento del árbol sintáctico: otro ejemplo de un 5-grama

de las fig. 4 y 5. La ruta marcada con flechas en negrita en la fig. 3 corresponde a un 5-grama sintáctico continuo. En este caso es: "*y di par de vueltas*".

A continuación presentaremos otro tipo de n-gramas sintácticos, donde se permiten las bifurcaciones.

III. N-GRAMAS SINTÁCTICOS NO-CONTINUOS

En esta sección vamos a presentar el complemento (o se puede considerar como una generalización) del concepto de n-gramas sintácticos continuos: n-gramas sintácticos no-continuos.

Como se puede observar en la sección anterior, la intuición detrás del concepto de n-gramas sintácticos continuos está relacionada principalmente con el hecho de que una secuencia

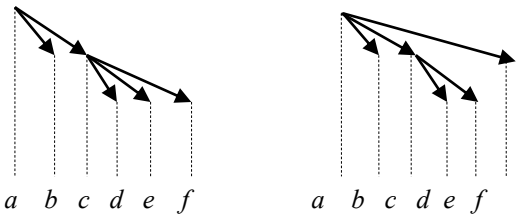

Fig. 6. Ejemplo de la ambigüedad en bifurcaciones

de palabras relacionadas puede ser considerada como tal, en su totalidad.

Sin embargo, existen otros conceptos lingüísticos interesantes, que no caben en el modelo de una secuencia unidimensional, por ejemplo, las valencias verbales (o patrones de rección) [19], [20]. Por ejemplo, el verbo *comprar* tiene los actantes: *quién, qué, de quién, por cuanto dinero*. Sería muy interesante tenerlos presentes al mismo tiempo en un n-grama. Sin embargo, tanto para el caso de n-gramas tradicionales, como de n-gramas sintácticos, todos esos componentes hubieran sido separados en n-gramas diferentes. Entonces, la intuición detrás del concepto de n-gramas sintácticos no-continuos es precisamente tratar de unir las palabras relacionadas semánticamente, aunque éstas no tengan una ruta continua, pero sí tengan alguna ruta que las conecte.

Es muy fácil dar una definición formal de n-gramas sintácticos no-continuos: son todos los sub-árboles de longitud *n* de un árbol sintáctico. En este sentido, digamos, los n-gramas sintácticos continuos se definen como todos los sub-árboles sin bifurcaciones de longitud *n* de un árbol sintáctico. Eso quiere decir que los n-gramas sintácticos continuos son un caso particular de n-gramas sintácticos no-continuos, al aplicar la definición propuesta. La longitud de un árbol es el número de arcos en este árbol, lo que corresponde al valor de *n* (en caso de los n-gramas).

Otro término que podemos proponer para denotar los n-gramas sintácticos no-continuos, es t-n-gramas (*tree n-grams*, en inglés) es decir, n-gramas de árboles[2].

Es cuestión de un trabajo futuro determinar qué tipo de n-gramas: continuos o no-continuos, son mejores para varios tipos de tareas de la lingüística computacional. Es posible que para algunos tipos de tareas son mejores unos, y para otros tipos de tareas, otros.

Cabe mencionar que el número de n-gramas sintácticos no-continuos es mayor que el de los n-gramas sintácticos continuos, dado que los últimos son un caso particular de los primeros.

El algoritmo de construcción (u obtención) de los n-gramas sintácticos no-continuos es relativamente sencillo. Para el nodo raíz hay que considerar todas las posibles combinaciones de sus hijos con el tamaño no mayor que *n*, y repetir este procedimiento de manera recursiva para cada nodo hijo. Y así de manera sucesiva hay que pasar por todos los nodos del árbol sintáctico.

Puede surgir la pregunta ¿y cómo representar a los n-gramas sintácticos no-continuos sin utilizar su representación gráfica? Cabe recordar que los n-gramas sintácticos continuos son simplemente secuencias de palabras, pero el caso de los n-gramas sintácticos no-continuos es diferente. Estamos proponiendo utilizar los siguientes convenios. Nótese que son convenios, por lo que pueden ser modificados en un futuro. Dentro de cada n-grama sintáctico no-continuo pueden existir unas partes continuas y una o varias bifurcaciones. Vamos a separar los elementos continuos de n-gramas con espacios en blanco —nada más, y en la parte de la bifurcación vamos a poner comas, además vamos a usar paréntesis para la parte de bifurcaciones, dado que posteriormente puede aparecer la ambigüedad de estructuras.

Dos ejemplos de los 5-gramas sintácticos no-continuos están representados en la fig. 4 y la fig. 5: "*y di par [un, de]*", "*y di [le, par, de_mala_gana]*". Nótese que los paréntesis y las comas ahora son partes de los n-gramas, pero eso de ninguna manera impide la identificación de los n-gramas sintácticos que son iguales.

La ambigüedad puede aparecer por ejemplo, en caso de que un n-grama tenga dos bifurcaciones y varios fragmentos continuos. Por ejemplo, el n-grama "*a [b, c [d, e, f]]*" y "*a [b, c [d, e], f]*" tiene el nodo *f* o bien como el tercer nodo debajo del nodo *c*, o bien como el tercer nodo debajo del nodo *a,* véase la fig. 6.

Ahora bien, tenemos dos posibilidades de manejar las partes con bifurcaciones: 1) tal como aparecen en el texto, que es la manera más natural, o 2) ordenarlos de alguna manera, por ejemplo, alfabéticamente. Esta última opción nos permite tomar en cuenta los cambios relacionados con el orden de palabras. Sin embargo, se necesitan más investigaciones para determinar cuál de las dos opciones es mejor y para qué tarea de PLN.

Otra posibilidad que nos gustaría mencionar es marcar directamente dentro de los n-gramas sintácticos no-continuos su profundidad. La intuición detrás de esta idea es que para algunos tipos de n-gramas sintácticos no-continuos puede ser importante su posición en el árbol sintáctico de la oración. En este caso la notación sería: "y_1 di_2 par_3 [un_4, de_4]", "y_1 di_2 [le_3, par_3, $de\_mala\_gana_3$]". Técnicamente, sería suficiente marcar el nivel de la primera palabra únicamente; podemos marcar los demás niveles también, pero no es estrictamente necesario.

A continuación vamos a considerar dos ejemplos de construcción de n-gramas sintácticos no-continuos, uno para el español y otro para el inglés, y los comparamos con los n-gramas sintácticos continuos.

[2] La sugerencia de A. Gelbukh es usar el término "t-gramas, árbol-gramas" (*tree grams*, *t-grams*, en inglés), sin embargo nos parece un poco mejor el término "n-gramas de árboles" (*tree n-grams*, *t-n-grams,* en inglés), dado que así se establece la relación del término propuesto con el concepto muy tradicional de los n-gramas. Del otro lado, una consideración a favor del término "t-grama" es su forma más simple.

IV. Ejemplo de construcción de n-gramas sintácticos no-continuos para el español

En esta sección vamos a presentar ejemplos de la construcción de n-gramas sintácticos continuos y no-continuos para el español. Tomemos la frase muestra:

Tomé el pingajo en mis manos y le di un par de vueltas de mala gana.

Para construir automáticamente los n-gramas sintácticos, es necesario aplicar antes un programa de análisis sintáctico automático, llamado en inglés *parser*. Para el español hemos utilizado el programa FreeLing [22], [23], que está disponible de manera gratuita.

El analizador sintáctico puede construir el árbol utilizando dos formatos: constituyentes y dependencias. El árbol de dependencias se presenta en la fig. 1, y el de constituyentes en la fig. 2. Ambos formatos tienen esencialmente la misma información de relaciones de palabras. Para los fines de construcción de los n-gramas sintácticos nos parece mejor utilizar las dependencias, porque la representación es más transparente. Sin embargo, de igual manera se puede utilizar el árbol de constituyentes.

Cabe mencionar que el analizador sintáctico primero realiza el análisis morfológico y la lematización. Como se puede observar, a cada palabra de la oración le corresponde su lema y la información gramatical, por ejemplo, "*Tomé tomar VMIS1S0*". Primero viene la palabra, después el lema, y al final la información gramatical.

La información gramatical utiliza el esquema de codificación EAGLES, que es el estándar de facto para el análisis morfológico automático del español. Por ejemplo, en la etiqueta VMIS1S0, la primera letra "V" corresponde al "verbo" ("N" hubiera sido un sustantivo, "A" un adjetivo, etc.), la "I" es el indicativo, la "S" significa el pasado, "1" es la primera persona, la otra letra "S" significa "singular". Como se puede observar, en cada posición se codifica un tipo específico de la información gramatical, y cada etiqueta tiene como máximo siete posiciones, de las cuales algunos pueden no usarse en algunos casos, por ejemplo, en caso de los sustantivos.

Primero presentamos los resultados de análisis automático de la oración utilizando el formalismo de constituyentes (fig. 2).

```
+coor-vb_[
 grup-verb_[
  +verb_[
   +(Tomé tomar VMIS1S0 -)
  ]
  sn_[
   espec-ms_[
    +j-ms_[
     +(el el DA0MS0 -)
    ]
   ]
   +grup-nom-ms_[
    +n-ms_[
     +(pingajo pingajo NCMS000 -)
    ]
   ]
  ]
  grup-sp_[
   +prep_[
    +(en en SPS00 -)
   ]
   sn_[
    espec-fp_[
     +pos-fp_[
      +(mis mi DP1CPS -)
     ]
    ]
    +grup-nom-fp_[
     +n-fp_[
      +(manos mano NCFP000 -)
     ]
    ]
   ]
  ]
 ]
 +(y y CC -)
 grup-verb_[
  patons_[
   +paton-s_[
    +(le le PP3CSD00 -)
   ]
  ]
  +grup-verb_[
   +verb_[
    +(di dar VMIS1S0 -)
   ]
  ]
  sn_[
   espec-ms_[
    +indef-ms_[
     +(un uno DI0MS0 -)
    ]
   ]
   +grup-nom-ms_[
    +n-ms_[
     +(par par NCMS000 -)
    ]
   ]
   sp-de_[
    +(de de SPS00 -)
    sn_[
     +grup-nom-fp_[
      +n-fp_[
       +(vueltas vuelta NCFP000 -)
      ]
     ]
    ]
   ]
  ]
  sadv_[
   +(de_mala_gana de_mala_gana RG -)
  ]
```

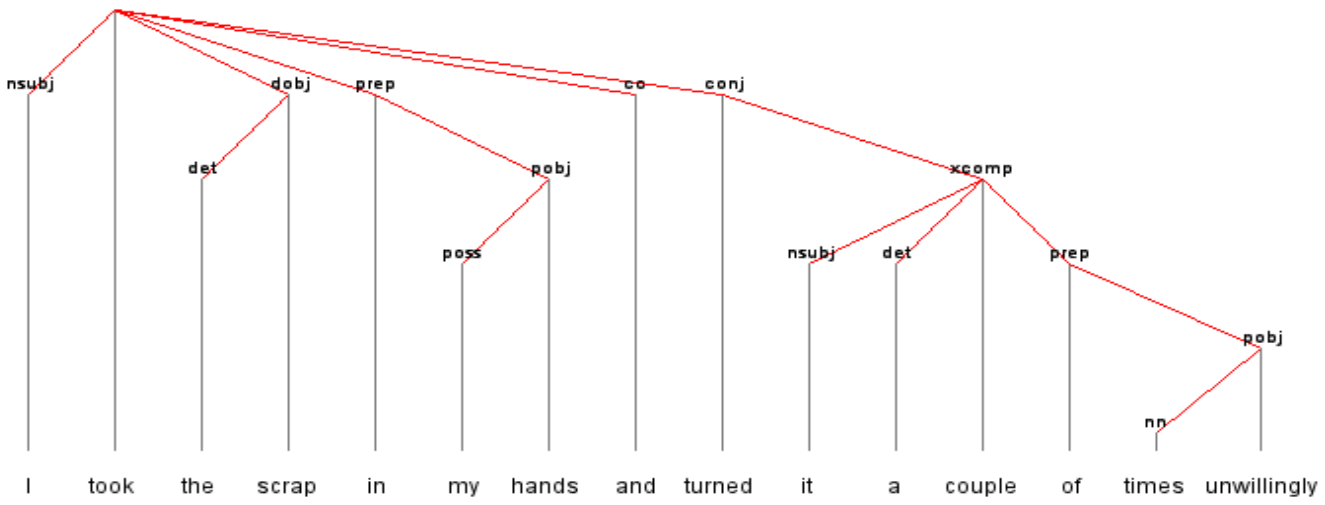

Fig. 7. El árbol sintáctico del ejemplo para el inglés

```
 ]
 F-term_[
  +(. . Fp -)
 ]
]
```

Información muy similar se presenta utilizando el formalismo de dependencias en la fig. 1.

```
coor-vb/top/(y y CC -) [
 grup-verb/co-v/(Tomé tomar VMIS1S0 -) [
  sn/dobj/(pingajo pingajo NCMS000 -) [
   espec-ms/espec/(el el DA0MS0 -)
  ]
  grup-sp/sp-obj/(en en SPS00 -) [
   sn/obj-prep/(manos mano NCFP000 -) [
    espec-fp/espec/(mis mi DP1CPS -)
   ]
  ]
 ]
 grup-verb/co-v/(di dar VMIS1S0 -) [
  patons/iobj/(le le PP3CSD00 -)
  sn/dobj/(par par NCMS000 -) [
   espec-ms/espec/(un uno DI0MS0 -)
   sp-de/sp-mod/(de de SPS00 -) [
    sn/obj-prep/(vueltas vuelta NCFP000 -)
   ]
  ]
  sadv/cc/(de_mala_gana de_mala_gana RG -)
 ]
 F-term/modnomatch/(. . Fp -)
]
```

Como ya mencionamos es más sencillo utilizar las dependencias, porque ellas prácticamente ya contienen los n-gramas sintácticos.

Se puede observar que las tres palabras "*de_mala_gana*" realmente son un solo adverbio.

Ahora bien, vamos a presentar los ejemplos de los n-gramas sintácticos extraídos. Primero presentamos los n-gramas sintácticos continuos.

Los bi-gramas sintácticos (en bi-gramas no hay diferencia entre bi-gramas continuos y no-continuos) son:

y tomé, tomé pingajo, pingajo el, tomé en, en manos, manos mis, y di, di le, di par, par un, par de, de vueltas, di de_mala_gana.

Los tri-gramas continuos son:

y tomé pingajo, y tomé en, tomé pingajo el, tomé en manos, en manos mis, y di le, y di par, y di de_mala_gana, di par un, di par de, par de vueltas.

Los 4-gramas continuos son:

y tomé pingajo el, y tomé en manos, tomé en manos mis, y di par un, y di par de, di par de vueltas.

No vamos a repetir los mismos elementos —elementos continuos—, aunque ellos también formen parte de los n-gramas sintácticos no-continuos. Nótese que en este caso tenemos que usar la notación propuesta para los n-gramas no-continuos para poder distinguirlos de otras configuraciones posibles. La notación forma parte de los n-gramas, no es algo adicional, es el propio n-grama. En los n-gramas sintácticos continuos las comas no son parte de n-grama, se pueden omitir poniendo un n-grama por línea. Entonces, como la coma ya es una parte de la notación, para ser más claros, ahora sí vamos a presentar un n-grama por línea. Entonces, los tri-gramas no-continuos nuevos en comparación con los n-gramas no-continuos son:

tomé [pingajo en]
di [le par]
di [le de_mala_gana]
di [par de_mala_gana]

par [un de]

Los 4-gramas no-continuos nuevos son:

tomé [pingajo el, en]
tomé [pingajo, en manos]
di [le, par un]
di [le, par de]
di [le, par, de_mala_gana]
di [par un, de_mala_gana]
di [par de, de_mala_gana]
par [un, de vueltas]

V. Ejemplo de construcción de n-gramas sintácticos no-continuos para el inglés

En esta sección vamos a analizar la construcción de n-gramas sintácticos para el inglés. Para simplificar la comparación con el español, tomaremos la traducción de la misma frase que en la sección anterior, la fig. 7, —en este caso la figura se generó de manera automática.

I took the scrap in my hands and turned it a couple of times unwillingly.

Si vamos utilizar el mismo analizador sintáctico, es decir, FreeLing, los resultados van a ser muy similares. Vamos a probar con otro analizador sintáctico para el inglés que también es bien conocido —analizador sintáctico de Stanford (Stanford *parser*) [24]. El árbol de constituyentes es como sigue:

```
(ROOT
 (S
  (NP (PRP I))
  (VP
   (VP (VBD took)
    (NP (DT the) (NN scrap))
    (PP (IN in)
     (NP (PRP$ my) (NNS hands))))
   (CC and)
   (VP (VBD turned)
    (S
     (NP (PRP it))
     (NP
      (NP (DT a) (NN couple))
      (PP (IN of)
       (NP (NNS times) (NN unwillingly)))))))
  (. .)))
```

En este *parser*, para el árbol de dependencias se usa una representación muy simple, pero expresiva: nombre de la relación, dos palabras (o sus *POS tags*, o lemas) junto con sus números en la oración. Primero se menciona la palabra principal y después la dependiente, es decir, el orden de las palabras es importante. Esta información permite construir el árbol sintáctico de manera única.

```
nsubj(took-2, I-1)
root(ROOT-0, took-2)
det(scrap-4, the-3)
dobj(took-2, scrap-4)
prep(took-2, in-5)
poss(hands-7, my-6)
pobj(in-5, hands-7)
cc(took-2, and-8)
conj(took-2, turned-9)
nsubj(couple-12, it-10)
det(couple-12, a-11)
xcomp(turned-9, couple-12)
prep(couple-12, of-13)
nn(unwillingly-15, times-14)
pobj(of-13, unwillingly-15)
```

Cabe mencionar que hemos desarrollado un programa en el lenguaje de programación *Python* que convierte el formato de dependencias de FreeLing al formato de dependencias de Stanford *parser*. Un problema relacionado con esa conversión consiste en que el formato de FreeLing no contiene los números de palabras en la oración, por lo que no se puede reconstruir el árbol de entrada en todos los casos, pero eso no afecta la construcción de los n-gramas sintácticos: se asigna un número consecutivo para identificar las palabras.

Se puede observar que aunque la frase es muy parecida, el otro *parser* aplicó otras reglas, específicamente, manejó de manera diferente la conjunción y cometió varios errores: por ejemplo, con *unwillingly*, relacionándolo con *of* y no con *turned;* con *it,* relacionándolo con *couple* y no con *turned*. Sin embargo, eso no afecta de manera conceptual nuestra discusión, ya que nuestra tarea no es mejorar algún *parser*. Ahora procedamos a la construcción de los n-gramas sintácticos continuos y no-continuos.

Los bi-gramas sintácticos (recordemos que no hay diferencia entre los bi-gramas sintácticos continuos y no-continuos) son:

took I, took scrap, scrap the, took in, in hands, hands my, took and, took turned, turned couple, couple it, couple a, couple of, of unwillingly, unwillingly times.

Los tri-gramas sintácticos continuos son:

took scrap the, took in hands, in hands my, took turned couple, turned couple it, turned couple a, turned couple of, couple of unwillingly, of unwillingly times.

Los 4-gramas sintácticos continuos son:

took in hands my, took turned couple it, took turned couple a, took turned couple of, turned couple of unwillingly, couple of unwillingly times.

Como en el ejemplo anterior no vamos a repetir los mismos elementos, aunque los n-gramas continuos forman parte de los n-gramas sintácticos no-continuos.

Los tri-gramas no-continuos nuevos son (puede ser que algunos de ellos son errores del *parser*, eso no afecta la idea propuesta dado que son errores de otro tipo, que se puede corregir mejorando el propio *parser*):

took [I, scrap]
took [I, in]
took [I, and]
took [I, turned]
took [scrap, in]
took [scrap, and]
took [scrap, turned]
took [in, and]
took [in, turned]
took [and, turned]
couple [it, a]
couple [it, of]
couple [a, of]

Los 4-gramas no-continuos nuevos son:

took [I, scrap the]
took [in, scrap the]
took [and, scrap the]
took [turned, scrap the]
took [I, in hands]
took [scrap, in hands]
took [and, in hands]
took [turned, in hands]
took [I, scrap, in]
took [I, scrap, and]
took [I, scrap, turned]
took [scrap, in, and]
took [scrap, in, turned]
took [in, and, turned]
couple[it, a, of]
couple[it, of unwillingly]
couple[a, of unwillingly]

Nótese que en este caso hemos tomado los elementos de los n-gramas sintácticos no-continuos en su orden de aparición en el texto. Como mencionamos, otra opción es ordenarlos de alguna manera, por ejemplo, alfabéticamente.

VI. Conclusiones

En este artículo hemos discutido cómo se realiza la investigación en la etapa moderna de la lingüística computacional, sobre todo relacionada con el uso de los métodos de aprendizaje automático [25], y hemos presentado una idea novedosa de n-gramas sintácticos no-continuos como posibles características en un modelo de espacio vectorial. Lo hemos comparado con nuestra idea presentada anteriormente de n-gramas sintácticos continuos.

Hemos considerado dos ejemplos para el español e inglés, brevemente hemos presentado el algoritmo de construcción de los n-gramas sintácticos no-continuos y hemos propuesto la notación formal de su representación. La notación —las comas y los paréntesis— es importante porque sin ella no hay manera de mantener la estructura —en principio multidimensional— de los n-gramas sintácticos no-continuos.

Se necesitan múltiples estudios experimentales para determinar qué parámetros de construcción de los n-gramas sintácticos no-continuos son mejores y para qué tipo de tareas existentes dentro de la lingüística computacional.

Agradecimientos

Trabajo realizado con el apoyo de gobierno de la Ciudad de México (proyecto ICYT-DF PICCO10-120), el apoyo parcial del gobierno de México (CONACYT, SNI) e Instituto Politécnico Nacional, México (proyectos SIP 20120418, 20131441, 20131702; COFAA), proyecto FP7-PEOPLE-2010-IRSES: Web Information Quality - Evaluation Initiative (WIQ-EI) European Commission project 269180. Agradezco a A. Gelbukh por su ayuda y sugerencias fructíferas.

Referencias

[1] I.A. Bolshakov, A. Gelbukh. *Computational linguistics: models, resources, applications*. 187 pp, 2004.

[2] C. Manning, H. Schütze, "Foundations of Statistical Natural Language Processing," MIT Press, Cambridge, MA, 1999.

[3] A. Gelbukh, G. Sidorov, "Procesamiento automático del español con enfoque en recursos léxicos grandes," IPN, 307 p., 2010.

[4] G. Sidorov, "N-gramas sintácticos y su uso en la lingüística computacional," *Vectores de investigación*, 6(6), 15 p., 2013.

[5] J.A. Reyes, A. Montes, J.G. González, D.E. Pinto, "Clasificación de roles semánticos usando características sintácticas, semánticas y contextuales," *Computación y sistemas*, 17(2), pp. 263–272, 2013.

[6] A. Gelbukh. "Using a semantic network for lexical and syntactical disambiguation," *Proc. CIC-97, Simposium Internacional de Computación*, Mexico, pp. 352–366, 1997.

[7] Y. Ledeneva, A. Gelbukh, R.A. García-Hernández. "Terms Derived from Frequent Sequences for Extractive Text Summarization," *Proc. CICLing 2008, Lecture Notes in Computer Science*, N 4919, pp. 593–604, 2008.

[8] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, "Syntactic Dependency-based N-grams as Classification Features," *LNAI*, 7630, pp. 1–11, 2012.

[9] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, "Syntactic Dependency-Based N-grams: More Evidence of Usefulness in Classification," *LNCS*, 7816 (Proc. of CICLing), pp. 13–24 (2013)

[10] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, "Syntactic N-grams as Machine Learning Features for Natural Language Processing," *Expert Systems with Applications*, in press, 8 p., 2013.

[11] H. Calvo, J.O. Juarez Gambino, A. Gelbukh, K. Inui. "Dependency Syntax Analysis using Grammar Induction and a Lexical Categories Precedence System," *Proc. of CICLing 2011, Lecture Notes in Computer Science*, N 6608, pp. 109–120 , 2011.

[12] M. Khalilov, J.A.R. Fonollosa, "N-gram-based Statistical Machine Translation versus Syntax Augmented Machine Translation: comparison and system combination," *Proceedings of the 12th Conference of the European Chapter of the ACL*, pp. 424–432, 2009.

[13] N. Habash, "The Use of a Structural N-gram Language Model in Generation-Heavy Hybrid Machine Translation," *LNCS*, 3123, pp. 61–69, 2004.

[14] A. Agarwal, F. Biads, K.R. McKeown, "Contextual Phrase-Level Polarity Analysis using Lexical Affect Scoring and Syntactic N-grams,"

Proceedings of the 12th Conference of the European Chapter of the ACL (EACL), pp. 24–32, 2009.

[15] A. Gelbukh. "Syntactic disambiguation with weighted extended subcategorization frames," *Proc. PACLING-99, Pacific Association for Computational Linguistics*, Waterloo, Canada, August 25–28, pp. 244–249, 1999.

[16] A. Gelbukh, I. Bolshakov, S. Galicia-Haro. "Statistics of parsing errors can help syntactic disambiguation," *Proc. CIC-98, Simposium Internacional de Computación*, November 11–13, Mexico City, pp. 405–515, 1998.

[17] S. Pado y M. Lapata, "Dependency-based construction of semantic space models," *Computational Linguistics*, 33(2), pp. 161–199, 2007.

[18] A. Gelbukh. "Natural language processing: Perspective of CIC-IPN," *International Conference on Advances in Computing, Communications and Informatics (ICACCI 2013)*, IEEE Conference Publications, pp. 2112–2121, 2013.

[19] S.N. Galicia-Haro, A. Gelbukh, I.A. Bolshakov. "Acquiring syntactic information for a government pattern dictionary from large text corpora," *IEEE International Workshop on Natural Language Processing and Knowledge Engineering, NLPKE 2001 at International IEEE SMC-2001 Conference: Systems, Man, and Cybernetics*. Tucson, USA, October 7–10, IEEE, pp. 536–542, 2001.

[20] A. Gelbukh, I. Bolshakov, S. Galicia Haro. "Automatic Learning of a Syntactical Government Patterns Dictionary from Web-Retrieved Texts," *Proc. International Conference on Automatic Learning and Discovery*, Carnegie Mellon University, Pittsburgh, PA, USA, June 11–13, pp. 261–267, 1998.

[21] M. Koppel, J. Schler, S. Argamon, "Authorship attribution in the wild," *Language Resources and Evaluation* 45(1), pp. 83–94, 2011.

[22] X. Carreras, I. Chao, L. Padró, M. Padró, "FreeLing: An Open-Source Suite of Language Analyzers," *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04),* 2004.

[23] L. Padró, E. Stanilovsky, "FreeLing 3.0: Towards Wider Multilinguality," *Proceedings of the Language Resources and Evaluation Conference (LREC 2012),* ELRA, Turkey, 2012.

[24] M.C. de Marneffe, B. MacCartney, C.D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," *Proc. of LREC*, 2006.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, 11(1), 2009.

More Effective Boilerplate Removal—the GoldMiner Algorithm

István Endrédy, Attila Novák

***Abstract*—The ever-increasing web is an important source for building large-scale corpora. However, dynamically generated web pages often contain much irrelevant and duplicated text, which impairs the quality of the corpus. To ensure the high quality of web-based corpora, a good boilerplate removal algorithm is needed to extract only the relevant content from web pages. In this article, we present an automatic text extraction procedure, GoldMiner, which by enhancing a previously published boilerplate removal algorithm, minimizes the occurrence of irrelevant duplicated content in corpora, and keeps the text more coherent than previous tools. The algorithm exploits similarities in the HTML structure of pages coming from the same domain. A new evaluation document set (CleanPortalEval) is also presented, which can demonstrate the power of boilerplate removal algorithms for web portal pages.**

***Index Terms*—Corpus building, boilerplate removal, the web as corpus.**

I. The Task

WHEN constructing corpora from web content, the extraction of relevant text from dynamically generated HTML pages is not a trivial task due to the great amount of irrelevant repeated text that needs to be identified and removed so that it does not compromise the quality of the corpus. This task, called boilerplate removal in the literature, consists of categorizing HTML content as valuable vs. irrelevant, filtering out menus, headers and footers, advertisements, and structure repeated on many pages.

In this paper, we present a boilerplate removal algorithm that removes irrelevant content from crawled content more effectively than previous tools. The structure of our paper is as follows. First, we present some tools that we used as baselines when evaluating the performance of our system. The algorithm implemented in one of these tools, jusText, is also used as part of our enhanced boilerplate removal algorithm. This is followed by the presentation of the enhanced system, called GoldMiner, and the evaluation of the results.

II. Existing Tools

In this section, some relevant boilerplate removal algorithms are reviewed, which are freely accessible and thus could be used as evaluation baselines. They contain good ideas, and the path of these good ideas are outlined in the following overview: methods often built on the result of the previous ones. We reimplemented some of these algorithms in C++, so that they can be evaluated in a fast and comfortable way.

Manuscript received on July 31, 2013; accepted for publication on September 30, 2013.

The authors are with the MTA-PPKE Language Technology Research Group and Pázmány Péter Catholic University, Faculty of Information Technology and Bionics, 50/a Práter street, 1083 Budapest, Hungary (e-mail: {endredy.istvan.gergely, novak.attila}@itk.ppke.hu).

A. The Body Text Extraction (BTE) Algorithm

The basic insights underlying the BTE algorithm [1] are the following:

1) the relevant part of the HTML content is usually a contiguous stretch,
2) the density of HTML tags is lower in it than in boilerplate content.

Based on these two assumptions, the algorithm performs a search for the longest stretch of text in which the number of intervening tags is minimal. The idea is simple, but the result is often wrong with the algorithm failing to extract the most relevant part of the content in situations where, contrary to the tag density assumption, it contains a segment with a higher tag-to-text ratio. This occurs, for example, if tables are included or advertisements interrupt the article. In this case, a significant part of the valuable content (or the whole) may be lost or replaced by entirely irrelevant content.

B. The Boilerpipe Algorithm

A merit of the boilerpipe [2] algorithm is that its authors demonstrated experimentally that boilerplate content can be identified effectively by using a good combination of simple text properties. They used an annotated training corpus of 500 documents (mainly Google news) to find the most effective feature combination. They tried to extract articles with the help of shallow text features, using 8-10 different feature combinations, and then they evaluated their results. In their experiments, a combination of word and link density features gave the best results (its F-measure was: 92%). Furthermore, the method is very fast and it needs no preprocessing. Both the training set and the tool can be downloaded.

C. The jusText Algorithm

The jusText algorithm [3] splits HTML content into paragraphs at block-level tags that are generally used to partition HTML content into logical units, such as `<p>`, `<td>`, `<h1>` etc. Using various features of these blocks of text such as the number of links (an idea from boilerpipe [2]), words and stopwords, the algorithm performs a rule-based

classification of the blocks using various thresholds and a language-dependent list of function words tagging each unit ‘good’, ‘almost good’, ‘bad’, or ‘too short’. The latter tag applies to units too short to categorize reliably. After initial classification, ‘almost good’ and ‘too short’ units surrounded by ‘good’ ones are reclassified as ‘good’. The text to be extracted consists of all units classified as ‘good’ in the final classification. The algorithm performs quite well even for extreme pages.

However, inspection of the corpus generated by using the jusText algorithm to filter crawled news portals revealed that many expressions that obviously come from a single article and should not occur more than once, like *The feeding-bottle is a potential source of hazard*, were still extremely strongly over-represented. Examples in Table I are from a corpus crawled from Hungarian news portals applying jusText as a boilerplate removal tool.

TABLE I
EXAMPLES OF PHRASES OVERREPRESENTED DUE TO INADEQUATE BOILERPLATE REMOVAL

Phrase	Occurr.
Utasi Árpi-szerű mesemondó. ‘Utasi Árpi-like storyteller.’	10,587
A cumisüveg potenciális veszélyforrás. ‘The feeding-bottle is a potential source of hazard.’	1,578
Obama amerikai elnök, ‘U.S. President Obama,’	292
etióp atléta: cseh jobbhátvéd ‘Ethiopian athlete: Czech right-back’	39,328
Barack Obama amerikai elnök ‘U.S. President Barack Obama’	2,372
George Bush amerikai elnök ‘U.S. President George Bush’	1,626

We found that the problem is caused primarily by jusText failing to eliminate leads of related and recommended articles and content coming from index pages containing only article headlines and leads. Leads and headlines of the set of current articles advertised on every article body page during the limited time span of the crawl were thus strongly overrepresented in the corpus.

D. JusText + Onion

JusText [3] was complemented with a post processing tool, called Onion (ONe Instance ONly), which is for removing near-duplicate paragraphs from the corpus. It generates a hash code for each sentence (n-gram of words), and only the first occurrence in the corpus is kept, others are dropped. It can be parametrized to drop whole documents or paragraphs containing duplicated parts. This method effectively decreases the ratio of duplicated content in the corpus, but it often decreases the coherence of the individual texts: they will not be continuous text any more: some parts may be missing from them.

Whether this is a problem or not depends on the aim of the corpus to be gathered. If the goal is just to have a huge collection of sentences, then the available algorithms may perform well enough, the best choice being most probably the jusText+Onion combo. But if it is considered a problem that the title and the lead of an article might be missing while it is attached to just another recent article, i.e. if the coherence of the text is important, then a new approach seems to be needed.

E. CleanEval

CleanEval [4] was a boilerplate remover competition held in 2007. The gold standard corpus used at that competition with a test set of 684 documents is available. The performance of new algorithms on this corpus can be evaluated using an improved evaluation script created by Evert [5]: it calculates precision, recall, F-score, true and false positives and negatives, etc. for the output of a given algorithm. This makes comparison to previously published tools possible.

The documents in the CleanEval corpus were prepared from English and Chinese web pages, which were selected at random: Google results for the following words were retrieved: *picture, extents, raised, events*. Annotators were asked to remove the boilerplate, and to identify the structure of the article (title, paragraphs, lists: using the h, p, l tags). This manually cleaned-up corpus is used as gold standard. The evaluation is based on Levenshtein edit distance [6], adapted by substituting ‘token’ for ‘character’. The calculated edit distance between each pair of cleaned files is divided by the file length: i.e. the percentage of all tokens from either of the two files that cannot be matched with a token in the other file.

III. THE GOLDMINER ALGORITHM

The problem of boilerplate removal from web pages generated by portal engines can be solved more efficiently if we step up to a level higher than that of individual web pages. As our first attempts at defining a good general procedure for identifying unwanted parts of pages were less successful than expected, we decided to take an optimistic stance and look for what is good instead of what is bad.

We based our approach on the following observations:

1) The relevant part of the HTML content is usually a contiguous stretch (see the BTE approach).
2) Within a web domain/subdomain, the internal structure (the HTML code) of dynamically generated pages generally contains common patterns that can help us identify relevant content.

The algorithm takes a sample of the pages of the domain/subdomain and tries to locate the common patterns in the HTML code within the sample that identify the beginning and the end of valuable content. For example, news portals typically advertise recent and related articles by displaying their headlines and leads next to the actual article. Although this usually seems to be relevant content to jusText, it is in fact just boilerplate content, like menus or advertisements, which has little or nothing to do with the actual article. Not filtering them out results in thousands of duplicates in the corpus.

```
<div class="search-filter">
    <input type="radio" value="up" name="shows" id="searchOptionsShow" /> <label for="searchOptionsShow">Up</label>
    <input type="radio" value="" name="shows" id="searchOptionsAll" /><label for="searchOptionsAll">All MSNBC</label>
</div>
<div class="search-wrap">
    <label class="screen-reader-text" for="s">Search</label>
    <input type="text" value="search" />
</div>
<div id="content" class="hfeed">
    <h1 class="entry-title">SEC may require more disclosure from corporations</h1>
    <div class="post-info"><span class="author vcard">Meredith Clark</span><div class="date published time">12:09 PM on 04/28/201

    <div class="entry-content">
    <p>Nearly a half a million people have signed a petition urging the SEC to require greater disclosure from corporations about
    Up with Steve Kornacki</em>, the panel discussed the effect the rule might have, its potential benefits and drawbacks, and th
    can) regulate money in politics.</p>
    <p>As the <em>New York Times</em> <a href="http://www.nytimes.com/2013/04/24/us/politics/sec-is-asked-to-make-companies-discl
    reported on Tuesday</a>, a broad coalition of activists, Democratic officials, and other interested parties has formed to pre
    to create a new rule that would require publicly traded corporations to disclose its political expenditures to its shareholde
    add your opinion to the discussion. How does one do it?</p>
    <p>If you want to add your voice to the conversation about regulating money in politics, corporate speech, and government reg
    from Sunday&#8217;s <em>Up with Steve Kornacki</em>, and send an email to <a href="mailto:rule-comments@sec.gov">rule-comment
    Number 4-637" and add your thoughts.</p>
    <p><em>Watch Up with Steve Kornacki every Saturday and Sunday at 8 AM.</em></p>
    <div id="jp-post-flair" class="sharedaddy sd-like-enabled sd-sharing-enabled"><h3 class="sd-title">Share this with friends</h
    <div class="sd-content"><ul><li class="share-twitter"><a href="..."><span>Twitter</span></a></li>
    <li class="share-facebook"><a href="..."><span>Facebook</span></a>
    [...]
    <h2>Discussions</h2>
    <span class="c-num">8 comments from 6 people</span>
    <a ...>jmlambion</a>   commented Apr 28, 2013
    <p>Per the FEC:</p>
    <p>"Foreign nationals may not make contributions in connection with any election--Federal, State or local. This prohibition d
    admitted for permanent residence in the United States (those who have "green cards")."</p>
    <p>As many Corporations have unlimited numbers of foreign investors how is contributing shareholder money not a violation of
```

Fig. 1. An example HTML content with unique and not unique paragraphs

Although, as we have seen, post-crawl de-duplication tools, like Onion, can remedy this situation by removing duplicate content, nothing guarantees, however, either that the only remaining instance of the duplicate content is the one that is at the right place or that all duplicates should be removed.

The algorithm learns the HTML tags identifying the beginning and the end of the article for each web domain/subdomain, and only content within this stretch of the page is kept. In addition, since it may still be the case that the body of the article is interrupted with advertisements or other boilerplate content at several points, it is submitted for further processing to the jusText boilerplate removal algorithm. An advantage of this solution is that text from pages with no article content (thematic index pages, tag clouds, search page results, etc.) will not be added to the corpus since the domain-specific HTML tag pattern is not present on them. The algorithm automatically discards the contents of these pages. However, all pages are, of course, still used as a source of URLs for the crawl.

A. A Detailed Description of the Algorithm

The first phase of the crawl of a domain is taking a sample, which is used to identify the domain-specific HTML tag pattern. The algorithm downloads a sample of some 100 pages, applying jusText categorization to each page, which breaks content into paragraphs and evaluates them. Repetitions of individual extracted paragraphs (identified as 'good' by jusText) over different pages in the sample are identified by the GoldMiner algorithm, and these paragraphs are reclassified as bad. Unique paragraphs remain classified as 'good'. Next, it finds the nearest common parent HTML tag of the good paragraphs in the DOM hierarchy on each page. At the end of the learning phase, the most frequent common good parent tag is identified as the winner.

We do not usually get optimal results, however, if the closing tag pair of this parent tag is simply chosen as the tag marking the end of the article. The span enclosed by the parent tag pair may contain bad paragraphs, too. In this case, the algorithm would not find the optimal cutting points. Therefore, it performs another search for the optimal starting and endpoint within the content of the previously selected tag, which may be a series of tags. With the selection of the cutting points, the learning phase for the domain is finished. As the URL domain is crawled afterwards, only the content between the domain-specific beginning and endpoint tag patterns is passed to the jusText boilerplate removal algorithm. Of course, pages used during the learning phase are also handled this way.

During the learning phase, GoldMiner uses only pages where the length of the extracted paragraphs reaches a threshold. Without using a threshold, it failed to learn the

optimal cutting points on some domains where thematic opening pages are more frequent than pages containing articles.

B. Illustration of the GoldMiner Algorithm

We present an example in Figure 1 to illustrate the algorithm.

In the learning phase, for every paragraph that was classified as 'good' by JusText, we check if it is unique or not among all pages downloaded from the same domain during the first phase of the crawl. Not unique paragraphs are reclassified as 'bad'. In this example, unique paragraphs are colored green, while those classified either by jusText as boilerplate or those occurring on other pages as well are colored red and marked by small red dotted arrows.

GoldMiner stores html patterns preceding and following green paragraphs. The fragment preceding the green span in the example is:

```
<label class="screen-reader-text"
     for="s">Search</label>
<input type="text" value="search"
     /></div>
<div id="content" class="hfeed">
```

The one following it is:

```
</em></p>
<div id="jp-post-flair"
   class="sharedaddy sd-like-enabled
   sd-sharing-enabled">
```

When the algorithm processed enough pages, it evaluates the stored patterns: it selects the most frequent uniquely identifiable pattern preceding and following the article body. In this example, the best pattern of enclosing tags is highlighted in blue and marked by bigger solid arrows. The configuration information learned for the given subdomain contains these html patterns. The html content of every page is trimmed using these patterns, only the content between the tags matching the patterns will be processed. Thus the otherwise unique content of comments (it also has green color on the picture as it is deemed 'good' by jusText) will be dropped from this page, it will not be considered part of the article.

IV. Evaluation

A. Results and Problems on the CleanEval Corpus

JusText and GoldMiner, with and without Onion post-filtering were tested on the CleanEval test set. As can be seen in Table II, Onion post-filtering increases precision while decreasing recall, which results in net reduction of the balanced F-score.

GoldMiner tries to learn the structure of pages characteristic of each (sub)domain, and applies jusText only to the part of the page that is expected to contain a relevant stretch of text. When comparing the results of GoldMiner with jusText on the CleanEval corpus, we do not get consistent improvement. This is not surprising, though, since this corpus does not contain more than just 3 to 4 pages from each domain, thus GoldMiner has no chance to learn anything relevant about the structure of the pages.

TABLE II
Results on CleanEval

	F-score	Precision	Recall
justText	93.61%	95.29%	91.99%
justText+Onion	93.24%	95.51%	91.08%
GoldMiner	93.40%	95.32%	91.55%
GoldMiner+Onion	93.08%	95.49%	90.78%
BoilerPipe	83.49%	95.15%	74.38%
BTE	91.09%	90.50%	91.68%

TABLE III
Results on CleanPortalEval

	F-score	Precision	Recall
justText	87.26%	78.82%	97.72%
justText+Onion	91.16%	86.48%	96.38%
GoldMiner	98.32%	98.50%	98.15%
GoldMiner+Onion	97.77%	98.48%	97.07%
BoilerPipe	90.68%	92.91%	88.56%
BTE	81.63%	71.20%	95.64%

It is worth mentioning that the corpus contains many torso articles after post-filtering with Onion: Onion often deletes paragraphs from the middle of the text. This often occurs with stereotypical sentences that occur many times in the corpus, like *Good Morning!* etc., and the text is fragmented without them. For example 127.txt in the CleanEval gold standard test set has this text:

```
<h>An open letter to KPLU
<p>To whom it may concern,
<p>Your radio feature by Kirsten
   Kendrick...
```

JusText keeps these paragraphs, but after post-filtering with Onion it looks like this:

```
<h>An open letter to KPLU
<p>Your radio feature by Kirsten
   Kendrick
```

The salutation, *"To whom it may concern,"* is missing. If we want to build a coherent text, not just a collection of independent sentences, the post-filtering performed by Onion may yield suboptimal results.

Moreover, the CleanEval gold standard sometimes does contain boilerplate (e.g. in 634.txt, the last <p> item) or broken words (e.g. 100-102.txt).[1] This, and the wish to demonstrate the power of GoldMiner inspired us to create a new evaluation set: CleanPortalEval, which contains more homogeneous sets of pages.

[1] Serge Sharoff's reaction (p. c.) to calling his attention to this fact: "Nobody is perfect."

TABLE IV
RESULTS ON 2 000 PAGES FROM VARIOUS NEWS PORTALS

Domain	Algorithm	Sentences	Uniq. snt.	%	Characters	Chr. in uniq.	%
origo.hu	BTE	60 682	33 269	54%	12 016 560	7 499 307	62%
	jusText	58 670	30 168	51%	8 425 059	4 901 528	58%
	GMiner	22 475	21 242	94%	3 076 288	3 051 376	99%
nol.hu	BTE	154 547	107 573	69%	24 292 755	13 544 130	55%
	jusText	186 727	128 782	68%	14 167 718	11 665 284	82%
	GMiner	162 674	123 716	76%	12 326 113	11 078 914	89%
index.hu	BTE	51 713	26 176	50%	5 756 176	4 061 697	70%
	jusText	40 970	29 223	71%	4 371 693	3 441 337	78%
	GMiner	13 062	11 887	91%	1 533 957	1 489 131	97%

V. CLEANPORTALEVAL

The wish to demonstrate that the approach implemented in GoldMiner is superior to a post-filtering approach for the task of extracting whole articles with minimally compromising the integrity of the texts prompted us to create a new gold standard document set. It contains several pages from the same domain (70 pages from 4 domains), thus it can be used to test the ability of various algorithms to clean pages generated by portal engines. Annotation in this gold standard corpus is similar to that of CleanEval: the output text is annotated using p, h, and l tags by human annotators. The CleanEval evaluation script can be applied to this test set without any changes (the corpus can be downloaded from `https://github.com/ppke-nlpg/CleanPortalEval`). The algorithms were tested on this document set, which yielded the following results, shown in Table III.

Note that, when testing on an appropriate test set that contains enough pages with similar structure, GoldMiner clearly outperforms its rivals both in terms of precision and recall. Applying Onion post-filtering to the GoldMiner output decreases not only recall but also precision in this case (test results can be downloaded from `https://github.com/ppke-nlpg/boilerplateResults`).

VI. RESULTS ON SOME PORTALS

Table IV shows the results of the GoldMiner algorithm compared with that of BTE and jusText on three Hungarian news portals: origo.hu, index.hu, nol.hu. The sample corpora quoted in Table IV were generated crawling just the first 2 000 pages from the domains above. Using GoldMiner, the ratio of duplicates in the corpus was reduced considerably compared to what other algorithms produced.

The results clearly show that the algorithm effectively reduces unnecessary duplication in these crawled corpora. Having not revised these pages manually, however, we have no estimate of how the different algorithms perform in terms of the amount/ratio of lost relevant content for these domains.

VII. CONCLUSION

In this paper, a new boilerplate removal algorithm, GoldMiner, was presented, which can eliminate boilerplate content from dynamically generated web pages in a more efficient way than similar available tools: it identifies recurring HTML tag patterns around relevant content characteristic of web pages coming from a given domain/subdomain. The algorithm preserves textual coherence better than the usual post-filtering de-duplication approach.

A new test document set was created to demonstrate its performance: previous gold standard corpora did not contain enough pages from the same domain for the approach to be applicable. The new gold standard set is called CleanPortalEval and it is open to the public.

ACKNOWLEDGMENTS

This research was partially supported by the project grants TÁMOP–4.2.1./B–11/2-KMR-2011-0002 and TÁMOP–4.2.2./B–10/1-2010-0014.

REFERENCES

[1] A. Finn, N. Kushmerick, and B. Smyth, "Fact or fiction: Content classification for digital libraries," in *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[2] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proceedings of the third ACM international conference on Web search and data mining*, ser. WSDM '10. New York, NY, USA: ACM, 2010, pp. 441–450. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718542

[3] J. Pomikálek, "Removing boilerplate and duplicate content from web corpora [online]," Ph.D. dissertation, Masarykova univerzita, Fakulta informatiky, 2011.

[4] M. Baroni, F. Chantree, A. Kilgarriff, and S. Sharoff, "Cleaneval: A competition for cleaning web pages," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, B. M. Nicoletta Calzolari, Khalid Choukri and D. Tapias, Eds. Marrakech, Morocco: European Language Resources Association (ELRA), 2008.

[5] S. Evert, "A lightweight and efficient tool for cleaning web pages," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, B. M. Nicoletta Calzolari, Khalid Choukri and D. Tapias, Eds. Marrakech, Morocco: European Language Resources Association (ELRA), 2008.

[6] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966, original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).

Journal Information and Instructions for Authors

I. Journal Information

Polibits is a half-yearly open-access research journal published since 1989 by the *Centro de Innovación y Desarrollo Tecnológico en Cómputo* (CIDETEC: Center of Innovation and Technological Development in Computing) of the *Instituto Politécnico Nacional* (IPN: National Polytechnic Institute), Mexico City, Mexico.

The journal has double-blind review procedure. It publishes papers in English and Spanish (with abstract in English). Publication has no cost for the authors.

A. Main Topics of Interest

The journal publishes research papers in all areas of computer science and computer engineering, with emphasis on applied research. The main topics of interest include, but are not limited to, the following:

- Artificial Intelligence
- Natural Language Processing
- Fuzzy Logic
- Computer Vision
- Multiagent Systems
- Bioinformatics
- Neural Networks
- Evolutionary Algorithms
- Knowledge Representation
- Expert Systems
- Intelligent Interfaces
- Multimedia and Virtual Reality
- Machine Learning
- Pattern Recognition
- Intelligent Tutoring Systems
- Semantic Web
- Robotics
- Geo-processing
- Database Systems
- Data Mining
- Software Engineering
- Web Design
- Compilers
- Formal Languages
- Operating Systems
- Distributed Systems
- Parallelism
- Real Time Systems
- Algorithm Theory
- Scientific Computing
- High-Performance Computing
- Networks and Connectivity
- Cryptography
- Informatics Security
- Digital Systems Design
- Digital Signal Processing
- Control Systems
- Virtual Instrumentation
- Computer Architectures

B. Indexing

The journal is listed in the list of excellence of the CONACYT (Mexican Ministry of Science) and indexed in the following international indices: LatIndex, SciELO, Periódica, e-revistas, and Cabell's Directories.

There are currently only two Mexican computer science journals recognized by the CONACYT in its list of excellence, *Polibits* being one of them.

II. Instructions for Authors

A. Submission

Papers ready for peer review are received through the Web submission system on www.easychair.org/conferences/?conf=polibits1; see also updated information on the web page of the journal, www.cidetec.ipn.mx/polibits.

The papers can be written in English or Spanish. In case of Spanish, author names, abstract, and keywords must be provided in both Spanish and English; in recent issues of the journal you can find examples of how they are formatted.

Only full papers are reviewed; abstracts are not considered as submissions. The review procedure is double-blind. Therefore, papers should be submitted without names and affiliations of the authors and without any other data that reveal the authors' identity.

For review, a PDF file is to be submitted. In case of acceptance, the authors will need to upload the source code of the paper, either Microsoft Word or LaTeX with all supplementary files necessary for compilation. Upon acceptance notification the authors receive further instructions on uploading the camera-ready source files.

Papers can be submitted at any moment; if accepted, the paper will be scheduled for inclusion in one of forthcoming issues, according to availability and the size of backlog. While we make every reasonable effort for fast review and publication, we cannot guarantee any specific time for this.

B. Format

The papers should be submitted in the format of the IEEE Transactions 8x11 2-column format, see http://www.ieee.org/publications_standards/publications/authors/author_templates.html. (while the journal uses this format for submissions, it is in no way affiliated with, or endorsed by, IEEE). The actual publication format differs from the one mentioned above; the papers will be adjusted by the editorial team.

There is no specific page limit: we welcome both short and long papers, provided that the quality and novelty of the paper adequately justifies its length. Usually the papers are between 10 and 20 pages; much shorter papers often do not offer sufficient detail to justify publication.

The editors keep the right to copyedit or modify the format and style of the final version of the paper if necessary.