21ST INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED17
21-25 AUGUST 2017, THE UNIVERSITY OF BRITISH COLUMBIA, VANCOUVER, CANADA

DETECTION AND SPLITTING OF CONSTRUCTS OF SAPPHIRE MODEL TO SUPPORT AUTOMATIC STRUCTURING OF ANALOGIES

Keshwani, Sonal; Chakrabarti, Amaresh
Indian Institute of Science, India

Abstract

The objective of this work is to structure a natural language description of analogies into a common causal language – which is chosen here to be SAPPhIRE model of causality. The motivation is to create a database of analogies that is structured so as to support focused search for analogies across the database. This should provide the benefit of utilizing the enormous data available on the Internet, while also providing relevant analogies to the designers as search results. This objective is achieved by implementing the following three steps: Firstly, detection of SAPPhIRE constructs in a document, achieved with an F-Measure of 0.834 using a text-classification approach; secondly, splitting sentences containing multiple SAPPhIRE constructs, achieved with an accuracy of 76.5% using a rule based approach; Thirdly, prediction of SAPPhIRE constructs for each text-input, implemented using the method proposed in literature. With these three steps, the time required to structure analogies into a common causal language can be reduced, thereby supporting population of the database and hence enabling designers in retrieving relevant analogies for novel idea generation.

Keywords: Computational design methods, Creativity, Bio-inspired design / biomimetics

Contact:
Sonal Keshwani
Indian Institute of Science
Centre for Product Design and Manufacturing
India
sonalkeshwani@gmail.com

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 4: Design Methods and Tools, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

Using analogies in designing supports novel idea generation. Analogical reasoning involves creation of similarity relations between familiar-domain (domain to which analogy is applied) and unfamiliar-domain (domain from where analogy is drawn) to draw further inferences. However, retrieving analogies is a challenging task for designers; the following are some of the major reasons: a) designer's knowledge is limited in unfamiliar domains; and b) there can be potentially innumerable analogies belonging to various domains. Therefore, researchers have proposed tools that support designers in retrieving analogies. These tools are based mainly on two approaches: a) tools that search a structured database of analogies (henceforth, denoted here as Approach-1); and b) tools that search natural-language documents for analogies (henceforth, denoted here as Approach-2). The tools based on Approach-1 have the benefit of focused search, but populating their database is a time intensive activity. In contrast, tools based on Approach-2 have the benefit of retrieving a large number of analogies for a given problem, but require methods with which to filter out irrelevant analogies searched. In order to reap the benefits of both the approaches and yet avoid their pitfalls, we propose to integrate these. We aim to do this by automatically converting an unstructured natural-language description of an analogy into the structured constructs of a common causal language. This should support automated population of a structured database of analogies. To achieve this, Keshwani and Chakrabarti (2017) used supervised text classification to label clauses into the constructs of SAPPhIRE model– a common causal language (See Section 4 for details). The current work builds upon the work of Keshwani and Chakrabarti (2017) and presents an implementation that uses the existing machine learning and natural language techniques to implement the proposed approach.

Analogies, in this work, belong to either natural- or artificial-systems; 'natural-systems' are those that occur in nature and are not human-made (for example, insects); 'artificial-systems' are taken as those that are human-made and belong to the domain of technical-products (for example, vacuum-cleaners).

2 LITERATURE REVIEW

This section reviews literature on the existing tools for retrieving analogies based on Approach-1 and Approach-2.

Using Approach-1, computational tools that search across a structured database of analogies have been proposed. For instance, Chakrabarti et al. (2005) developed IDEA-INSPIRE, the database of which is structured using SAPPhIRE model of causality; Similarly, Vattam et al. (2011) developed DANE, in which the database is structured using SBF model. AskNature uses a biomimicry taxonomy of engineering functions to categorize the natural-systems in its database (Deldin and Schuknecht, 2014).

Using Approach-2, tools that search for analogies in natural-language documents have been proposed. For instance, Hacco and Shu (2002) searched for potential analogies in the index and glossary of biology text book by incorporating the use of Wordnet part-of-speech tags with a keyword search method. Using Wordnet troponym and hypernym relations, word frequency and word collocation, they identified biologically meaningful keywords (Chiu and Shu, 2007), and later on translated functional terms of functional basis into biologically meaningful keywords (Cheong et al., 2011). Vandevenne et al. (2013) developed a webcrawler that continuously searches the Internet for biological strategies to update its knowledge base. Apart from these, research efforts have also been made to search for analogies from patent space (Verhaegen et al., 2011; Fu et al., 2013; Murphy et al., 2014).

The intensity of research efforts using both the approaches indicates the perceived potential for supporting design-by-analogy for novelty. However, each of the approaches have some benefits and limitations. The tools based on Approach-1 have the benefit of focused search (Sarkar et al., 2008) but populating their database is a tedious activity. Similarly, the tools based on Approach-2 have the benefit of large database for retrieving a large number of analogies for a given problem, but they require methods with which to filter out irrelevant analogies searched (Glier et al., 2014).

Therefore, in this work, we propose a third approach that integrates the above two approaches in such a way that the benefits of both are retained, while avoiding the limitations of both. We argue that this integration can be achieved by automatically converting natural-language descriptions of analogies into structured entries in a database.

As far as we are aware, the closest work to this has been reported by Vandevenne et al. (2015) who used supervised classification to predict the category of a natural-system in the biomimicry taxonomy used

in AskNature. According to them, this will aid in the automatic population of the database of AskNature. Our work is distinct from theirs in the following ways: a) while Vandevenne et al. (2015) have classified natural-systems into the categories of taxonomy of Ask Nature to populate the database, we have attempted to structure, both natural- and artificial-systems into a common causal language to populate the database; b) their work is restricted to natural-systems that are classified through the biomimicry taxonomy; our work on the other hand uses the features of a common causal language to classify analogies irrespective of the domain of the system; and c) due to the inherent nature of the common causal language used in our work, we have also been able to capture causal relations of an analogy – something that is not captured by Vandevenne et al. (2015). Another study that is somewhat similar to the work presented here is reported by Kaiser et al. (2013) who used keywords describing the constructs –'functions of technical-systems', 'properties of technical-systems' and 'environmental influences' – to search for relevant analogies in the database of PubMed. Our work is different from their work in the sense that the common causal language used here has seven constructs (SAPPhIRE model of causality) that provide a more detailed description than the three constructs used by them; SAPPhIRE model has been empirically tested to be representative of how designers proceed when they naturally carry out design (Srinivasan and Chakrabarti, 2010); further the model is developed by integrating multiple, fragmented models of functional reasoning, thereby providing a comprehensive, integrated description of causality in functional systems (Chakrabarti et al., 2005). This greater detail, more integral representation, and empirical grounding, we argue, should aid in more focused search, thereby helping retrieve more relevant analogies.

3 RESEARCH QUESTION AND OBJECTIVE

From the above literature review, the following research question has been asked: how to support designers in retrieving analogies that are relevant to the design problem? The following is the associated research objective: to automatically structure analogies into a common causal language so as to create a structured entry in the database.

4 RELATED WORK

Keshwani and Chakrabarti (2017) proposed to structure natural-language descriptions of analogies into SAPPhIRE model. Using supervised text-classification approach, they automatically predicted SAPPhIRE constructs of the *text-inputs* within a natural language description of an analogy. According to them, a text-input is the sentence or a clause that includes only one SAPPhIRE construct. Using Linear SVM classifier, precision, recall, F-Measure and accuracy achieved by them were 0.71, 0.69, 0.70 and 0.70±0.08 respectively. However, their work had the following limitations: a) they manually processed the documents to remove the sentences that did not belong to any SAPPhIRE construct; and b) they manually converted sentences into text-inputs, so as to predict the SAPPhIRE construct for each of these sentences. This work extends the above work of Keshwani and Chakrabarti (2017) by automating these two manual steps, thereby automating the process of structuring of an analogy.

As this work builds upon the work of Keshwani and Chakrabarti (2017) who used SAPPhIRE model, in this work also, we have used SAPPhIRE model to structure analogies. Chakrabarti et al. (2005) developed this model for supporting design by providing causal descriptions of biological- and technical-systems as stimuli for inspiring ideation for designers searching for solutions to design-problems. The acronym SAPPhIRE stands for its constructs State-Action-Part-Phenomenon-Input-oRgan-Effect. The digits 1-7 in parenthesis in Figure 1 denote the hierarchy in SAPPhIRE abstraction levels. Ranjan et al. (2012) describe this model as follows:

Components and interfaces that comprise an entity and its surroundings (parts) have some properties and conditions (organs). When the entity and the surrounding are not in equilibrium with each other, there is transfer of a physical quantity in the form of a material, energy or signal (input) across the boundary of the entity. This physical quantity, in combination with relevant properties and conditions, together activate a principle (physical-effect). This principle is responsible for an interaction (physical-phenomenon) between the entity and the surrounding. The interaction between the entity and the surrounding changes various properties of the entity and the surrounding (state-change). The change in properties can be interpreted at a higher level of abstraction (action). This model of causality built upon the above constructs and links is called SAPPhIRE model.

5 RESEARCH APPROACH

This section presents the methodology for achieving the objective mentioned in Section 3. The steps of this methodology are detailed in Sections 5.1 – 5.3 and illustrated in Figure 2. They are implemented in Python using Textblob (Loria, 2014), Scikit Learn (Pedregosa et al., 2011) and NLTK (Bird, 2006).

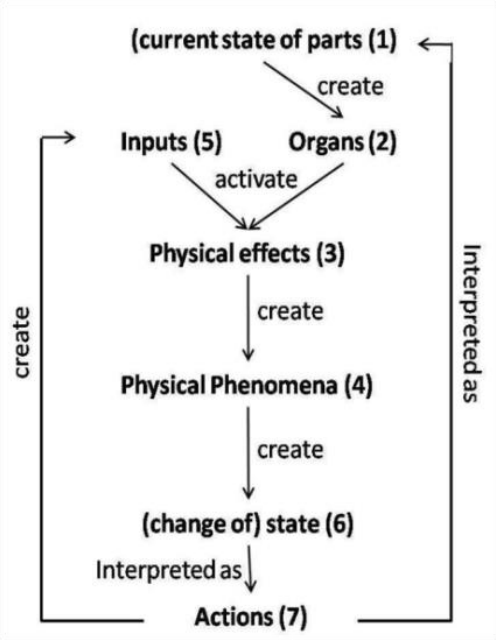

Figure 1. SAPPhIRE model of causality (Source: Chakrabarti et al., 2005)

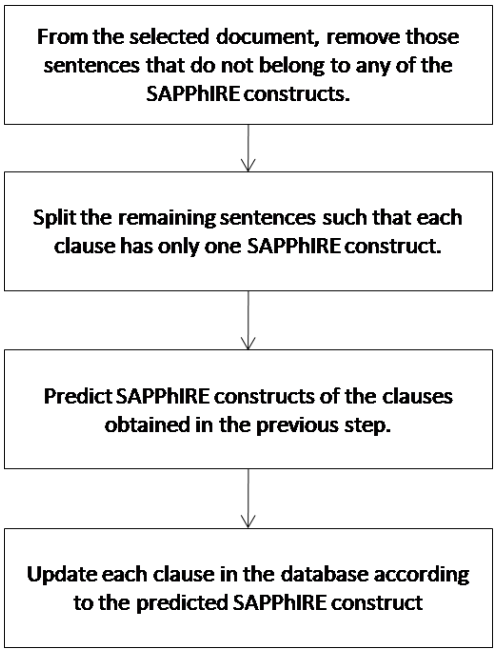

Figure 2. Steps for structuring an analogy into SAPPhIRE constructs

5.1 Removal of sentences that do not belong to any SAPPhIRE construct ('outliers') from a document

In order to automatically structure a natural-language description of an analogy from a document into its SAPPhIRE-model, it is important to first remove those sentences that do not belong to any construct of the SAPPhIRE model, for example, "In this edition of HowStuffWorks, we will discuss the history and technology behind these popular writing instruments so that you can understand them completely!" (Brain, 2001). Henceforth, such sentences will be termed here as 'outliers'; and the sentences that belong to any of the SAPPhIRE constructs will be termed as 'inliers'. In order to automatically distinguish outliers from inliers, the following alternative methods were tried: a) outliers were detected using One-class SVM classification; b) probability estimates of outliers (in the opinion of the authors) obtained using Logistic Regression classifier were studied to observe if there exists a threshold probability for the outliers; c) 'no-class' was introduced as the eighth class for predicting outliers in the classification approach used by Keshwani and Chakrabarti (2017); and d) supervised binary classification of sentences into 'inliers' and 'outliers' using Linear SVM classifier. In this section, supervised binary classification (method (d) above) which gave the best result, is detailed.

In machine learning, *classification* is the task of choosing the correct *class label* for a given input (Bird et al., 2009). This task of classification is performed by a function called *classifier*. Supervised classification involves training a classifier on *training data,* and then testing its performance on *testing data*. Both training and testing data contain texts, along with the correct class label for each text (Bird et al., 2009). In this work, the two class labels are 'outlier' and 'inlier'. Table-1 shows the distribution of training and testing data used in this work.

Table 1. Training and testing data for classification of sentences as outliers or inliers

Data	Outliers	Inliers
Training Sentences	338	776
Testing Sentences (30-35% of training data)	103	264

Sentences in the training and the testing data were searched from the following sources: a) How stuff works? (Brain, 2001); b) Physical laws and effects (Hix and Alley, 1958); c) Idea Inspire database (Chakrabarti et al., 2005); and d) Extraordinary animals: an encyclopedia of curious and unusual animals (Piper, 2007).

5.1.1 Pre-processing and feature extraction steps

The sentences in the training data were first pre-processed. The pre-processing steps included the following: a) word tokenization; b) part-of-speech (POS) tagging; c) lemmatization; d) stop-word removal and e) reduction of variability in sentences. These steps are explained below.

Word tokenization is the task of cutting strings into words and *part-of-speech tagging* is classifying these words into their parts-of-speech (Bird et al., 2009). These two tasks were implemented here using TextBlob. *Lemmatization* is the task of stripping off any affixes from a word and ensuring that the resulting form of the word is a known word in a dictionary (Bird et al., 2009). It was implemented using Wordnet Lemmatizer. *Stopword removal* is the task of removing high frequency words with little lexical content for example, 'is', 'the', 'and', etc. (Bird et al., 2009). This list was created using Stopwords Corpus and the authors' experience of using SAPPhIRE model.

Next, we studied approximately 80 documents from Brain (2001) and Piper (2007); from these, we identified topics associated with a product that should not be included in a SAPPhIRE model. These topics were as follows: product history, advantages or disadvantages, explanation of diagram, questions and emotions, symbolic meanings, product classifications, URLs, references and introductory and concluding statements in a document. Based on those topics, eleven clusters of words were created; these clusters, in the opinion of the authors, could occur in those topics and could be helpful in distinguishing outliers from inliers. The clusters for outliers are enlisted in Table 2. Similar clusters were required to be created for inliers. Fifteen such clusters for inliers, identified by Keshwani and Chakrabarti (2017), were considered in this work. Some examples of these clusters are enlisted in Table 3. The words, characters, and phrases in these clusters were searched for across the training data, and replaced with the words marked in bold at the start of each cluster. This helped to *reduce the variability* in the sentences. Pre-processed sentences were then searched for *features* (described below) that distinguished an inlier from an outlier.

Features highlight differences among the classes, using which a classifier predicts a label of a given input. According to Bird et al. (2009), "feature extractors are built through a process of trial-and-error, guided by intuitions about what information is relevant to the problem". The following features were selected, after trial and error, based on the performance of classification: a) suffixes- '-nce', '-ment', '-ion'; b) word unigrams; c) word bigrams; d) word trigrams; e) pos-tag bigrams; and f) pos-tag trigrams. For each sentence in the training data, its features along with the actual-label of the sentence were passed to Linear SVM classifier, in order to train the classifier. SVM is one of the best approaches used in the area of discriminative classification. In this work, selection of N-grams as features for classification created a very high dimensional and sparse vector space. For this type of large, sparse, vector space, a linear SVM classifier provides competitive and often better accuracy than its nonlinear counterparts (Yuan et al., 2012). Therefore, Linear SVM has been used here for classification.

Table 2. Clusters of words likely to occur in outliers

Cluster 1	[**outlier,** modern, year, chronicle, time, history, ……]
Cluster 2	[**outlier**, advantage, problem, reward, trouble, ……..]
Cluster 3	[**outlier**, intend, envisage, guess, ideate, imagine, fantasize, …….]
Cluster 4	[**outlier,** label, show, portray, depict, picture, outline, ……]
Cluster 5	[**outlier,** lesson, illustrate, instance, exercise, exemplify, ……]
Cluster 6	[**outlier,** interpret, feel, experience, learn, feeling, bet, …….]
Cluster 7	[**outlier,** thought, scheme, idea, theme, rationale, strategy, ….]
Cluster 8	[**outlier,** simple, basic, elementary, introductory, uncomplicated, ……]
Cluster 9	[**outlier,** we, you, our, let's, their, your, themselves, …….]
Cluster 10	[**outlier,** ?, !, @, :]
Cluster 11	[**outlier,** as shown, as follows, include:, this article, let us, …..]

Table 3. Examples of clusters that may occur in inliers (Keshwani and Chakrabarti, 2017)

Cluster 1	[**inlier**, purpose, objective, aim, goal, result, outcome, function, ……....]
Cluster 2	[**inlier**, change, decrease,...]
Cluster 3	[**inlier**, provide, inflow,...]
Cluster 4	[**inlier**, provide, hooked up, pump in,…]
Cluster 5	[**inlier**, connected, joined,….]

5.1.2 Results of predicting outliers

Once the classifier was trained, the above steps were implemented on the test data to predict if the sentence was an inlier or an outlier. For each sentence in the test data, the predictions made by the classifier were then matched against the labels assigned by the researchers (called here as 'actual-labels'). Based on the agreement between the actual- and the predicted-labels, the performance of the classifier was evaluated, as shown in Table 4. Precision determines out of the total documents (here, sentences) that were classified as X, how many were actually X. Recall determines the number of documents (here, sentences) belonging to class X, that were correctly classified. F-Measure is the harmonic mean of precision and recall (Martin and Jurafsky, 2000).
Therefore, the classifier trained in this step can now be used for predicting outlier sentences from a given document describing an analogy.

Table 4. Results of classification using Linear SVM classifier

	Precision	**Recall**	**F-Measure**
Outliers	0.762	0.747	0.759
Inliers	0.901	0.908	0.904
Average	0.831	0.827	0.829

5.*2* Splitting of inliers into text-inputs

Once the outliers are removed from a document, it is important to split the remaining sentences into text-inputs (see Section 4 for definition). This is because, a sentence may include multiple SAPPhIRE constructs, for example, "With the fence gone (organ), the bolt can slide freely (phenomena) past and the safe can be opened (action)".
This section describes rules that have been used here to split a sentence into text-inputs. These rules have been framed, after observing 97 sentences (four complete documents of 'How Stuff Works?') and 262 clauses, by the authors who had more than five years of experience in using SAPPhIRE model. Existing parsers could have been used here. However, we have used a rule based approach; this is because, we wanted to split the sentences according to the rules that are specific to the constructs of SAPPhIRE model. For each rule, an example has been provided. Text-inputs in these examples are denoted by { }. These rules are mentioned in the decreasing order of the priority in which they will be applied for splitting.

- **Rule 1:** Do not split a sentence if the words in the given Cluster A are present in the sentence. This rule has been framed for the 'Part' construct of SAPPhIRE model.
  Example: {Shaft of the motor is connected to the blades of the fan.} This sentence describes a configuration of parts – the blades of a fan and its shaft. Therefore, whole sentence is a part level description which should not be split.
  Cluster A: [connect, attach, engage, fasten, fix, mount, include, compose, touch, touches, consist, composed, made, join].
- **Rule 2:** Do not split a sentence if the words in the given Cluster B1 are followed by words in the given Cluster B2. This rule has been framed for Organ and Input SAPPhIRE constructs. It was observed by the authors that Input and Organ constructs have the following syntax in the natural language description of analogy: {A phrase involving words in Cluster B1}{words in Cluster B2}{condition}. Such sentences, if split at words in Cluster B2, will lose the notion of 'condition' – which is an essential feature of Organ and Input. Therefore, such sentences should not be split.
  Example: {It is important that the battery remains charged as long as possible.}

Cluster B1: [availability, require, need, necessary, requisite, important, essential, compulsory, mandatory, crucial, significant, affect, presence, absence, critical, sufficient, enough, determine, lack].
Cluster B2: [to, for, of, that].

- **Rule 3:** Do not split a sentence if the words in the given Cluster C are present in the sentence. This rule has been framed for Physical_Effect, Organ and Physical_Phenomena constructs of the SAPPhIRE model. It was observed by the authors that description of a Physical_Effect, Organ, or Physical_Phenomena generally contain several clauses. However, since all the clauses together describe a single construct, the sentence should not be split.
  Example: {The amount of drag force that air exerts on dust particles is proportional to the diameter of the particle and to the difference in velocities between the particle and air.}
  Cluster C: [constant, ratio, number, proportion, sum, product, proportional, equal, inverse, inversely, vary, defined, known, called].
- **Rule 4:** Split a sentence if the phrases in the given Cluster D are present in the sentence. This rule is framed for separating Part and Action constructs from each other. It was observed that certain sentences have the following syntax: {Noun Phrase}{Auxiliary verbs like is, 'are' 'has been', etc.}{Phrases in Cluster D}{Action}. Therefore, such sentences should be split as shown in the following example: {A silencer}{is}{*used to*}{reduce the amount of noise and visible muzzle flash generated by firing.}
  Cluster D: [used to, used for, useful for].
- **Rule 5.1:** Split a sentence at punctuation marks in Cluster E.
  Example: {If you push a wedge against an object},{it will push the object to the left or right.}
  Cluster E: [, , ; , :," ,"].
- **Rule 5.2** (Exception for Rule 5.1): Do not split a sentence if the part-of-speech tags of preceding and succeeding words of the punctuation (,) are the same.
  Example: {a hook is generally coupled with a loop (NN), eye (NN) or hollow area.} Here word 'loop' and 'eye' both are nouns separated by a comma (,).
- **Rule 5.3** (Exception for Rule 5.1): Do not split if the word succeeding punctuation (',') is either a determiner, or an adjective or an adverb followed by a word with same part-of-speech tag as that of the word preceding conjunction.
  Example: {Included in the carbohydrate group are natural sweeteners (NNS), refined (JJ) sugars (NNS), starches, cellulose, and various other substances.}
  Rules 5.1 - 5.3 are framed according to the rules of English grammar and can be used to split any two SAPPhIRE constructs if they are separated by a punctuation mark in Cluster E.
- **Rule 6:** Split a sentence at verb except light verbs present in Cluster F. A light verb is a verb that has little semantic content of its own. This rule can be used to split any two SAPPhIRE constructs if they are separated by a verb.
  Example: {Rotation of fan}, {*creates* swirling motion in air molecules.}
  Cluster F = [has, have, had, do, did, done, is, are, was, were, has been, have been, had been].
- **Rule 7.1:** Split a sentence at conjunctions.
  Example: {In a car engine *therefore*}{all of the fuel is loaded into the cylinder}.
- **Rule 7.2** (Exception for Rule 7.1): Do not split if the part-of-speech tags of preceding and succeeding words for the conjunctions –'and', 'or', are same.
  Example: {because they are simple (JJ) and sturdy (JJ).}
- **Rule 7.3** (Exception for Rule 7.1): Do not split if the word succeeding conjunction ('and', 'or') is either a determiner, or an adjective or an adverb followed by a word with same part-of-speech tag as that of the word preceding conjunction.
  Example: {a motor consists of a stator (NN) and a rotor (NN).}
  Rules 7.1 – 7.3 are framed according to the rules of English grammar and can be used to split any two SAPPhIRE constructs if they are separated by a conjunction.

Please note that while Rules 1-4 are framed specifically according to the patterns followed by SAPPhIRE constructs as observed by the authors, Rules 5.1-7.3 are general rules for splitting a sentence into clauses. Therefore, Rules 1-4 are given higher priority over Rules 5.1-7.3 for splitting a sentence into clauses. Here pos tag ''NN' is singular noun, NNS is plural noun, and 'JJ' is adjective according to Penn Tree Bank's (Marcus et al., 1993) list of POS tags.

5.2.1 Results of splitting a sentence into text-inputs

In order to assess the accuracy of splitting done by Rules 1-7 created in this work, a test was undertaken. 80 sentences (two complete documents from 'How stuff works?') were considered. The text-inputs produced by using these rules were then checked by the first author. A score of 1 was awarded if the splitting was correct; 0 was awarded if the splitting was incorrect according to the author. It was found the out of 80 sentences, 61(76.25%) sentences were correctly split.

5.3 Prediction of SAPPhIRE construct of each text-input

Once the sentences are split into text-inputs, the SAPPhIRE construct of each of them can be predicted using the previous work by Keshwani and Chakrabarti (2017), see Section 4 for detail.
Once the SAPPhIRE constructs of clauses are predicted, these clauses can be updated into a database.

6 DISCUSSION

In this section, we provide a comparison of the results of evaluation of our tool with similar tools reported in literature and also discuss the reasons behind the errors that occurred in a) removing outliers; and b) splitting of sentences into SAPPhIRE constructs.

6.1 Comparison of proposed support with similar supports

Vandevanne et al. (2015) proposed to automatically populate the database of AskNature by identifying the category to which a biological system belongs in the biomimicry taxonomy. They reported classification precision of 62.5% for top ten classes in the taxonomy. For us, the challenge is different from those of Vandevanne et al. (2015). In order to populate the database of Idea Inspire (Chakrabarti et al., 2005), we aim to extract the SAPPhIRE models from natural language description. As described in Section 5, we have achieved this in the following three steps: a) removal of outliers with an F-Measure of 0.834; b) splitting sentences into clauses with accuracy of 76.5%; and c) prediction of SAPPhIRE constructs with F-Measure of 0.70. As these are sequential steps, the evaluation measures of the preceding step will influence the evaluation measures of the succeeding steps. Calculation of overall accuracy of the database is a part of future work.

We now explain the significance of the first step - outlier detection which is unique to our work. In general, a document will contain both relevant and irrelevant (inliers and outliers in this work) types of sentences with respect to the description of a particular phenomena under consideration. These sentences need to be segregated to effectively extract causal relations and to reduce the reading effort of the designer. With outlier removal proposed in this work, this process of segregation has been automated. This step was not implemented by Vandevanne et al.(2015) as their reference corpus consisted of manually cleaned up biological strategies that were previously available in AskNature. This manual cleaning requires lot of effort. According to Deldin and Schuknecht (2014), *"AskNature's original data set represents a huge amount of human labor …. Individuals continue to generate additional content for AskNature…".* We hope that this step can support, to some extent, researchers who create strategy pages in AskNature.

6.2 Errors observed

We now discuss the errors that occurred in the removal of outliers. Out of the 103 sentences labeled as outliers by the authors, 26 were predicted as inliers by the classifier; out of the 264 sentences labeled as inliers by the authors, 24 were predicted as outliers. Therefore, total 50 cases of errors were observed out of 367 sentences. These errors occurred because of the following reasons: a) seven cases were observed where the sentence was composed of multiple of clauses – where some clauses were outliers and some were inliers, for example, 'since then, scientists have greatly improved upon Volta's original design (outlier) to create batteries made from a variety of materials that come in a multitude of sizes (inlier).'; b) twelve cases of errors were observed because certain words that indicated SAPPhIRE constructs were also present in the outliers, for example, 'the ancient Egyptians had a great *need* for air conditioning.'; and c) for the remaining 31 cases, the reason behind errors in classification was not understood, for example, 'new principal cells are produced from groups of undifferentiated cells at the base of the epithelium.' Clearly, this sentence is an inlier but was predicted as outlier by the classifier.

We now discuss the errors that occurred in splitting sentences into SAPPhIRE constructs. Total 19 cases of errors were observed, the following are the main reasons: a) two occurred because punctuation (.) was used in abbreviation instead of period, for example, {All functions in a modern engine are controlled by the ECM communicating with an elaborate set of sensors measuring everything from *R.P.M* .}{engine coolant and oil temperatures and even engine position *(i.e.*} {*T.}{D.}{C.*} {*)* .}; b) six errors occurred due to applicability of multiple rules, for example, {Piston rings *provide* a sliding seal between the outer edge of the piston *and*} {the inner edge of the cylinder.}; c) four due to incorrect POS tagging done by the pos tagger, for example, {The crankshaft turns the piston 's up (NN) and} {*down (IN)* motion into circular motion.}; d) three errors occurred due to missing words in the list clusters of created by the authors; and e) the remaining four occurred due to miscellaneous reasons.

7 CONCLUSION

In this work, we have attempted to automatically structure analogies using the SAPPhIRE model of causality. In the first step, the document was cleaned up by removing those sentences that did not belong to any construct of the SAPPhIRE model. Using text-classification approach, average precision, recall and F-Measure achieved for this step were 0.830, 0.837 and 0.834 respectively. In the second step, multiple SAPPhIRE constructs existing within a single sentence were split so as to correctly predict the SAPPhIRE constructs existing within a sentence. Using rule based approach, the accuracy achieved in this step was 76.5%. The third step - prediction of the SAPPhIRE construct for each text-input, was implemented using the method proposed by Keshwani and Chakrabarti (2017). These three steps together will aid in the creation of a structured database of analogies that can be used for searching analogies at the seven abstraction levels of the SAPPhIRE model, using simple, combination and complex search (Sarkar et al., 2008). We hope that this implementation will be able to address the problem of scalability associated with computational tools based on structured databases. The following are the main contributions of this work:

- This work demonstrates a new approach for retrieval of analogies i.e. converting natural language description of analogies into a structured database.
- The clusters identified in this work to distinguish sentences involving SAPPhIRE constructs from the remaining sentences are general. Therefore, these clusters can be used by other researchers working on building repositories for retrieval of analogies.
- As SAPPhIRE model is domain independent, this approach can be adapted for mining patents.
- Automatic creation SAPPhIRE model may aid other design tasks such as automatic evaluation of novelty of concepts which requires comparison of newly generated concepts with those of the existing concepts at various levels of abstraction (Srinivasan and Chakrabarti, 2010).

This work has the following limitations:

- Initially we intended to collect the data from Brain (2001). However, it was observed that the documents in Brain (2001) included State_Change and Physical_Effect constructs in lesser numbers than those of the other SAPPhIRE constructs. Therefore, we searched for these two constructs from two other sources - Idea Inspire database (Chakrabarti, et al., 2005) and Hix and Alley (1958). The data collected from these three sources was mostly related to artificial-systems. To incorporate data from natural-systems, Piper (2007) was referred. Therefore, data for this work was collected from four different sources having more than four authors. It is likely that the difference in the writing styles of these authors would have introduced variability in the data, which would have reduced the performance of our support.
- Rule based approach has been used for splitting sentences into SAPPhIRE constructs. Even though the rules were created by the authors who had experience in using SAPPhIRE model, the problem of scalability exists. Further work involves using parsing techniques for the same.
- Description of a complex system requires creation of multi-instance SAPPhIRE model and identification of input – output relations among the instances. The current implementation cannot differentiate among the constructs of multiple instances of the SAPPhIRE model. This challenge needs to be addressed in future.

Future work also includes replacing clusters with word2vec model, population and testing of the database, and testing the influence of the populated database on novelty of the designs produced.

REFERENCES

Bird, S., (2006), "NLTK: The Natural Language Toolkit." *In Proceedings of the COLING/ACL on Interactive presentation sessions*, Association for Computational Linguistics Sydney, pp. 69-72.

Bird, S., Klein, E. and Loper, E., (2009), *Natural language processing with Python*. O'Reilly Media, Inc.

Chakrabarti, A., Sarkar, P., Leelavathamma, B. and Nataraju, B.S. (2005), "A Functional Representation for Aiding Biomimetic and Artificial Inspiration of New Ideas." *AIEDAM*, Vol.19 No.02, pp.113-132.

Deldin, J.M. and Schuknecht, M., (2014), "The AskNature Database: Enabling Solutions In Biomimetic Design". In Goel A.K, McAdams D.A., Stone R.B., (Ed.) *Biologically Inspired Design*, Springer, London, pp. (17-27). DOI: 10.1007/978-1-4471-5248-4_2

Cheong, H., Chiu, I., Shu, L.H., Stone, R.B. and McAdams, D.A. (2011), "Biologically Meaningful Keywords for Functional Terms of the Functional Basis", *Journal of Mechanical Design*, Vol. 133 No. 2, pp.(021007). DOI: 10.1115/1.4003249

Chiu, I. and Shu, L.H., (2007), "Biomimetic Design Through Natural Language Analysis to Facilitate Cross-Domain Information Retrieval", *AI EDAM*, Vol. 21 No. 01, pp.45-59. DOI: 10.1017/S0890060407070138

Fu, K., Cagan, J., Kotovsky, K. and Wood, K. (2013), "Discovering Structure in Design Databases Through Functional and Surface Based Mapping", *Journal of Mechanical Design*, Vol.135 No.3, p.031006. DOI: 10.1115/1.4023484

Glier, M.W., McAdams, D.A. and Linsey, J.S. (2014), "Exploring Automated Text Classification to Improve Keyword Corpus Search Results for Bioinspired Design", *Journal of Mechanical Design*, Vol. 136 No. 11, pp.111103. DOI: 10.1115/1.4028167

Hacco, E. and Shu, L.H. (2002), "Biomimetic Concept Generation Applied to Design for Remanufacture", *ASME IDETC/CIE* (pp. 239-246).

Brain, M., (2001), *How Stuff Works*, Discovery Communications, Inc, New York.

Hix, C. F, and Robert P. A., (1958), *Physical Laws and Effects*. John Wiley.

Kaiser, M.K., Hashemi Farzaneh, H. and Lindemann, U. (2013), "Bioscrabble—Extraction of Biological Analogies Out of Large Text Sources", *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*., Portugal, 19-22 September, 2013, Springer.

Keshwani, S., and Chakrabarti, A. (2017), "Towards automatic classification of description of analogies into SAPPhIRE constructs", *ICoRD*, Guwahati, 7-9 January 2017, Springer.

Loria, S., (2014), "TextBlob: simplified text processing." Secondary TextBlob: Simplified Text Processing.

Marcus, M.P., Marcinkiewicz, M.A. and Santorini, B. (1993), "Building a large annotated corpus of English: The Penn Treebank", *Computational Linguistics*, Vol. 19 No.2, pp.313-330.

Martin, J.H. and Jurafsky, D., (2000), *Speech and Language Processing*. International Edition, 710.

Murphy, J., Fu, K., Otto, K., Yang, M., Jensen, D. and Wood, K. (2014), "Function Based Design-By-Analogy: a Functional Vector Approach to Analogical Search", *Journal of Mechanical Design*, Vol. 136 No. 10, pp.101102. DOI: 10.1115/1.4028093

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., (2011), "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research*, Vol. 12, pp. 2825-2830.

Piper, R., (2007), *Extraordinary Animals: An Encyclopedia of Curious and Unusual Animals*. Greenwood Publishing Group.

PubMed, https://www.ncbi.nlm.nih.gov/pubmed/ (15 May 2016).

Ranjan, B.S.C., Srinivasan, V., Chakrabarti, A. (2012), "An Extended, Integrated Model Of Designing", In Horváth. I., A, Behrendt, M., Rusak Z. (Eds), *TMCE*, Karlsruhe, Germany, 7-11 May, 2012.

Sarkar, P., Phaneendra, S. and Chakrabarti, A. (2008), "Developing Engineering Products Using Inspiration From Nature", *Journal of Computing and Information Science in Engineering*, Vol. 8 No. 3, pp.031001. DOI: 10.1115/1.2956995

Srinivasan, V. and Chakrabarti, A. (2010), "Investigating novelty–outcome relationships in engineering design", *AIEDAM*, Vol. 24 No. 02, pp.161-178. DOI: https://doi.org/10.1017/S089006041000003X

Vandevenne, D., Caicedo, J., Verhaegen, P.A., Dewulf, S. and Duflou, J.R. (2013), "Webcrawling for a biological strategy corpus to support biologically-inspired design", *CIRP Design 2012*, Bangalore, Springer, London pp. 83-92.

Vattam, S., Wiltgen, B., Helms, M., Goel, A.K. and Yen, J., (2011), DANE: Fostering Creativity In and Through Biologically Inspired Design, *Design Creativity 2010*, Springer, London, pp. 115-122.

Verhaegen, P.A., D'hondt, J., Vandevenne, D., Dewulf, S. and Duflou, J.R. (2011), "Identifying Candidates for Design-By-Analogy", *Computers in Industry*, Vol. 62 No. 4, pp. 446-459. DOI: 10.1016/j.compind.2010.12.007

Vandevenne, D., Verhaegen, P.A., Dewulf, S. and Duflou, J.R. (2015), "A Scalable Approach for Ideation in Biologically Inspired Design", *AIEDAM*, Vol. 29 No. 01, pp.19-31. DOI: 10.1017/S0890060414000122

Yuan, G.X., Ho, C.H., Lin, C.J. (2012), "Recent advances of large-scale linear classification". *IEEE* 100, no. 9, pp. 2584-2603.