IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI), Singapore, 2013, pp. 92-99

BrainGene: computational creativity algorithm that invents novel interesting names

Maciej Pilichowski
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Toruń, Poland
Email: bluedzins@wp.pl

Włodzisław Duch
Department of Informatics
Faculty of Physics, Astronomy and Informatics
Nicolaus Copernicus University, Toruń, Poland
Email: Google W. Duch

***Abstract*—Human-level intelligence implies creativity, not only on the grand scale, but primarily in the everyday activity, such as understanding intentions, behavior, and invention of new words. Psychological models of creativity have some support in experimental cognitive psychology, but computational models of creative processes are quite rare. This paper presents a model of creative processes behind invention of novel words related to description of products and services.**

I. Introduction

As with many terms related to human behavior precise definition of creativity is not easy. Karl Popper has stated that creativity should not even be measured, because of its “ divine spark nature" [1]. Science values order, logic and deduction, but great ideas were not born in that way. At a later stage, when the ideas have reached mature state, logical and ordered presentation is important, but at the very moment of their birth situation is quite different. On the grand scale creativity is associated with scientific law discoveries, paradigm shifts and inventions [2], and often illustrated with such example as Kepler’s laws or the discovery of ring structure of benzene molecule [3]. No wonder that research on creativity drew attention mainly of philosophers, psychologists, and educationalists who describe stages of creative ways of solving problems and design tests useful in estimating the level of creativity, and its relation to other concepts such as intelligence or personality.

Creativity is usually understood as the ability to create novel, useful and surprising things [4]. It is also a part of everyday human thinking, understanding language expressions, neologisms, and behavior of other people. Therefore creativity must be one of the most important aspects of the human-level intelligence.

Perhaps the simplest domain where creativity may be experimentally investigated and also simulated with sufficient accuracy to test “Blind Variation, Selective Retention" (BVSR) hypothesis (described in the next section) is creation of novel words [5], [6]. In this paper development of our *Brain-Gene* algorithm is described. It may be considered as an implementation of the BVRS hypothesis restricted to creation of novel words describing names of products, web sites or company names. Several international companies specialize in such services. Although the algorithm has been inspired by the neurocognitive principles mentioned above it will be presented in the statistical, rather than neural framework. In our opinion computational intelligence (CI) is a branch of science that searches for solutions of problems for which effective algorithms either do not exist or are not known [7]. From this point of view statistical or probabilistic approaches are as much part of CI as any soft computing algorithm.

In the next section a broader background on creativity is presented, followed by the algorithm description and results of computational experiments.

II. Creativity

There have been so many attempts to define creativity that it should be evident that it may not be possible to find a satisfactory definition that agrees with all human intuitions. Etymologically Latin word “cogito" means “to shake together” [8], stressing the importance of associations. One of the questions about creativity is its nature – is creative thinking “just" a form of thinking, shaking thoughts together until they click, or is it a qualitatively different process? Psychologist advocate such “creative thinking techniques” as brainstorming or lateral thinking, but with rather limited success [1]. However, there is little evidence for their effectiveness ([9]) – the results are judged as good because they are compared to results of boring corporate way of working. Altshuller [10] was even more skeptical noting that brainstorming – developed over the period of 12 years since 1977, with expense of 4 million dollars – is usually a team effort, so its effectiveness should be measured counting how many man-hours were spent [11].

Many hypotheses are related to the source of the creativity. Can creativity be a conscious process or is it done by unconscious processing? Hadamard [8], who was a famous mathematician, was one of the first to reject the view that unconscious processes may control only mechanical tasks. He claimed that unconsciousness not only can create but also choose solutions, so that the whole creative process should be located within unconsciousness. There is a lot of support for this view in experimental psychology [12], [13]. We do not have access to most processes that run in the brain, so even if there was an algorithm for creative thinking it might be difficult to find it. Goldenberg et al. [9] have already shown how systematic use of templates leads to ideas that are on par with some of the best creative ideas in advertising. Comparative tests of human and computer creativity are limited to strict procedure performed better than “unbounded” one (without creation procedure).

Boden [14] proposed to understand creativity as the ability to assemble old ideas and judge degree of novelty by the probability of combination of their components. She has distinguished between novelty at the personal and at the historical level. The former is called P-creativity (ideas that are novel for the person who created them), and the latter one H-creativity (for ideas created for the first time in history). This distinction is hard to maintain in practice, as similar ideas are born independently when the time is ready from the seeds of the past, and it may not be possible to say what is P-creativity and what is H-creativity.

Altshuller and his colleagues working on the TRIZ algorithm for facilitating inventions analyzed many patents [10] and proposed more subtle distinction of the degree of the novelty and thus required creativity. They implicitly assumed that the patents are granted only for real inventions, i.e. they did not have to decide if the invention was novel. In this case creativity is needed for:

1) basic enhancement – 32% of analyzed patents are at this level: for example, after designing the diver suit, the next natural step was adding ability to change the size of the „shoe” by introducing the set of lead shoe toes of various size;
2) improvement – 45% of patents improve an existing idea (within the same technology branch), for example adding middle layer of metal for welding, enhancing quality of connection;
3) major improvement – 18% of patents lead to a radical change using solutions taken from other technology branches;
4) radical breakthrough – 4% of patents go beyond current paradigm, for example changing electromechanic switch in favor of the semiconductor one;
5) discovery – only 1% of patents are based on new discoveries going beyond existing knowledge [11].

Discovery is very rare and is a privilege of a very few people. Enhancements and improvements of designs are already made by genetic programming and template-based techniques. For example, Koza has summarized dozens of patented results that improve upon previous solutions [15].

Goertzel has presented an interesting view on creativity in terms of sub-selves [3], or psychological entities that develop as the result of creative activity and separation from environment. In the brain various complex quasi-stable complex processes responsible for psychological sub-selves may form. They lack strict integration, Goertzel sees creative sub-self as separate from judging or commenting sub-self that cannot be active at the same time. Too much or too little criticism is an obstacle for creativity, and thus such processes like critic and creator should be viewed as partially autonomous, yet at the same time collaborating with each other. His second observation states that the frequency of oscillation between creative and ordinary sub-self is different in various subdomains. For example, in modern art the critic can intervene only at the end of the process, while in mathematics actions of critic have to intertwine with activity of creator quite often. The oscillation frequency should be adaptive. The feeling of creative inspiration is identified with awareness of emergent pattern watched from “inside”. Finally creativity implies the ability of big reconstruction of sub-selves, reorganization of memory, sometimes destroying some sub-selves.

Psychological inspirations may be useful, but real understanding of processes behind creative thinking should link them to spontaneous processes emerging from the brain neurodynamics. In recent years some understanding of such processes and their relation to insight, imagination, intuition and creativity has emerged [16], [17], and this understanding is now slowly translated into computational models [18], [5], [19], [6]. These models are based on several assumptions:

- trained neural networks provide search space of internal states that reflect past experience of the system, constraining possible neural activations;
- priming of neural networks due to the recent history of activations increases probability of cooperation between neurons encoding fragments of distributed representations;
- blind variation is a combinatorial processes linking partial activations of primed neural circuits into larger meaningful chunks;
- the most active chunks, relevant in a given context, win the competition filtering out less active chunks;
- results appear as spontaneous thoughts, solutions to the problem, intentions for the next action.

Filtering of the most active chunks is based on enhancement of their transient activity by feedback from currently primed fragments of neural networks, providing context in which associations and expectations operate, and leading to emotional arousal if a strong enhancement of overall network activity occurs. Such enhancement is strongest when larger chunks, or many smaller chunks, of active subnetworks synchronize with each other, creating a single larger whole through the feedback loops that sustain their activity for some time. Such putative brain mechanisms may underpin Campbell’s psychological model of creativity [20], called “Blind Variation, Selective Retention" (BVSR). In a long review of this approach to creative thought Simonton [21] shows that abstract combinatorial models are supported by experimental cognitive psychology, and may be related to personality traits, developmental factors, and social contexts. Computational models of BVSR or other processes related to thinking are quite difficult to make, as representations of concepts in the brain involves many areas distributed around the brain [22], [23], [24]. Although some attempts have been made to implement the meaning of general concepts in neural networks [25] without explicit model of perception recreation of natural associations and thus meaning may not be feasible.

III. BrainGene algorithm

The task to be solved is: given a description of some object (product, internet site, service, organization, invention) create a neologism that is easy to remember and that reflects some qualities strongly associated with the object itself. From the end-user perspective *BrainGene* program takes only a few steps to get the results, i.e. user inputs text containing short description or a list of words that set the topic, and optionally specifies the number of iterations for searching related words in the *WordNet* [26] database. The extended set of words is then used to obtain from dictionaries all derivations, inflections, irregular forms of verbs, etc. The result of this process is a set of priming words $\mathcal{W}$ that represents immediate associations potentially arising in the brain of intelligent person. For example, the input word “go” is transformed into “go, goes, went” and so on. The result set of words is considered as priming set. It should be matched to the set of all words, representing the background knowledge of the linguistic system.

General sketch of the ideas behind the *BrainGene* is as follows. The number of occurrences of every possible word $w \in \mathcal{W}$ in a large corpus is counted. Next, every word is split into n-grams, and those are put in the matrix – the n-gram determines the position within the matrix, while its occurrence count determines the value of the matrix element. The way this value is calculated depends on the settings of the program: it could be raw counts, binary values (0 for absent, 1 for present), probabilities and so on. When done, values in entire matrix are recomputed – depending on settings, it can be a normalization of the matrix, or just a row, or clipping the values to given range (typically 0–1), etc. This ends the procedure of calculating first matrix representing general knowledge of statistical relations in a given language.

The priming set is now used to build the second matrix, representing active subpart of the general knowledge. After the estimation of likelihood that some n-grams are more probable than another are calculated for both sets of words – those taken from the dictionary and those from the priming set – the two matrices are added (alternatively maximum values may be selected for each cell). This is based on assumption that priming increases probability of well-established associations in an additive way. One could play with rescaling parameters combining the two matrices, but we have not investigated such possibility yet. The final matrix is used as core data for creating new words from combinations of the most active n-grams. These steps are presented in the diagram below (I).

The newly created word should pass several filters: the word rank should not be too low, it should not resemble closely any word in the dictionary (one can also check if this is not a substring of some other, potentially offending words, using the multilingual.sensegates.com service), and it should be associated with the priming words through overlapping n-grams. Novel words fulfilling these conditions are added to the result set, and when this set grows too large words with lowest ranks are removed from it.

As the dictionary and as the corpus we have used data from *Google’s Web 1T 5-gram* [27]. This corpus contains raw data and it is not possible to filter words by, for example, their source, making the algorithm more specialized in specific context areas. Some preprocessing is necessary in order to eliminate slang, Spanglish or simply misspelled words. For this purpose *LRAGR* [28] and *SCOWL* [29] dictionaries have been used. There is another quite surprising problem with *Google Web 1T 5-gram* corpus, namely its shallowness: over 40% of words which are present in regular English dictionaries do not exist in *Web 1T*. This is probably due to the fact regular dictionaries contain many very rare or archaic English words. They should anyway have marginal influence on the novel words produced by *BrainGene* algorithm, as probabilities associated with such words are very low.

The data from *Web 1T* corpus are normalized before they are used, the occurrences are recalculated in the logarithmic scale to avoid the effect of overrepresentation of some words. For example the count of "the" in *Web 1T* is about 23 billion, while the second most used word – "o" – is counted 13 billion times. On the other hand the count of “viscoses" is merely about 200.

Priming

Creation of novel pseudo-words related to a given subject relies on priming, adding new words associated with the input (priming) words. However, most semantic associations (for example, “health" $\rightarrow$ “running") are beyond capabilities of the program, so the user has to build much broader input, explicitly adding semantic associations in order to include them in combinatorial processing.

Another problem is related to word sense disambiguation. Let’s consider word “light". It can mean for example “having little weight", or it can mean “electromagnetic radiation of any wavelength that travels..." (both definitions come from Merriam-Webster dictionary). However the program is not aware which meanings related to other words should be added creating the result set and which should be omitted. When advertising a product it is more useful to tie “possibilities" with “many" (positive association), rather than “problems" (negative associations). Currently the system does not contain any information about positive or negative interpretations of the words, so it is recommended to use only the words of one kind, like “bound" and “less" or such as “more" and “energy".

The system does not possess any knowledge about the world, the only facts it knows are the words, but not their meanings. In effect any word is for the program as good as any other. The system does not hesitate to use obscene words (with interesting results). Since it is not appropriate to create such words for commercial purposes, and we will have to wait for reliable real world knowledge database, efforts are undergoing to apply negative priming mechanism, which would allow to use selected words but such usage would be penalized.

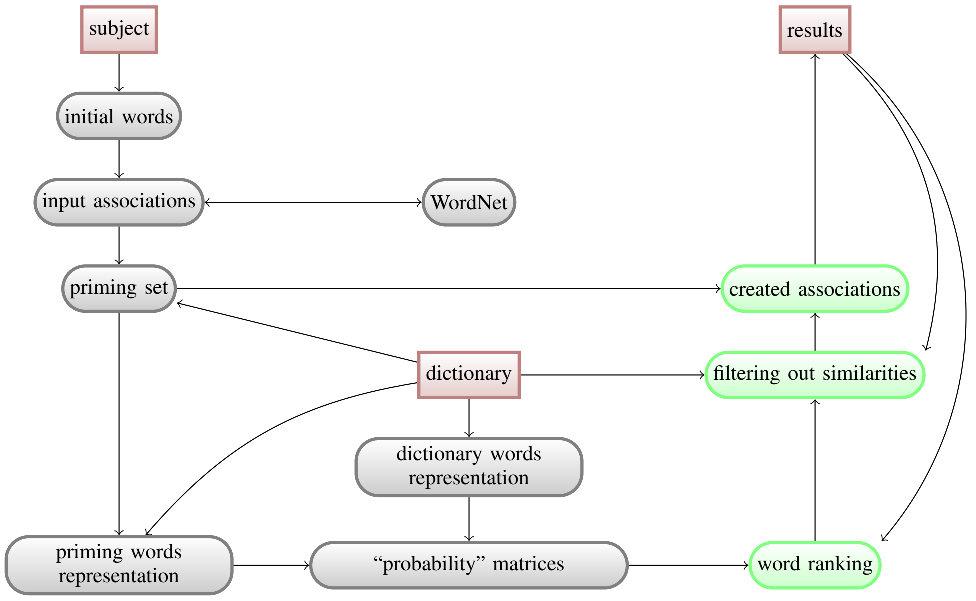

Table I
WORKFLOW OF THE DATA, INPUT AND OUTPUT ARE MARKED WITH RED RECTANGLES, PROCESSING STEPS – WITH OVALS.

Imitations

Compound words like “bodyguard", “brainstorm" or “airmail" are highly ranked by the system. When testing the program with the subject set to “aviation” there was created interesting result: “jetmail". Linguistically it is correct, but it was too good to be true, so when analyzing how it was created it appeared it was created thanks to a kind of plagiarism. The priming set contained word "jet", and the dictionary contained word "airmail". The first one let created word pass through association check, the second one helped gain high score (because building the new word was really just a replacement).

To avoid such short path in building words all compound words were banned from the dictionary. Thanks to that, there is guarantee that the created words result from deeper computation, not naive exchanging of two sub-words. However, such exchanging of word parts may lead to interesting results too.

WORD RANKING FUNCTION

We can semi-supervise the algorithm, teach it linguistically correct word construction by controlling the word score that should estimate the quality of word construction. Such score should estimate the degree to which the word is similar to some training examples. It has some analogy to the process of human learning from experience rather than rules (babies start to speak long before they learn the difference between nouns and verbs). The brain unconsciously synthesizes the rules from incoming facts (in early stages of development, due to insufficient data, often incorrectly [30]). This hidden knowledge is often called “intuition" ([31]).

The same mechanism can be used for priming. There is a possibility to compute associations between created words and priming set, but the computing time in such case is significant. Instead one can increase the pressure from the priming set (over-representation) – this is not as precise as checking associations, but there is no time penalty for such approach.

The word rank function $Q'(w)$ is defined as follows:

$$Q'(w) = \prod_{i=0}^{|ng(T(Z(w)))|-1} q(ng(T(Z(w)))[i])$$

where Z denotes transcription of given word, T is composition of word transformations, q is the likelihood of n-gram, and function $ng(w)$ splits the word w into a sequence of portions of text of length $N_{ng}+1$, each, starting after previous one with the shift of S_{ng} symbols (usually characters). N_{ng}, S_{ng} and G_{ng} (gap within n-gram) are parameters for entire program and are entered by the user.

For example, assuming that each character is a distinct symbol splitting the word “sound" into n-grams gives the result:

$$ng(\text{“sound"}) = \langle\text{“son", “oud"}\rangle \quad : N_{ng} = 2, S_{ng} = 1, G_{ng} = 1$$

The $ng(w)$ function is defined as below:

$$\begin{aligned} ng(w) = \langle & w[0 : N_{ng} - 1] \cdot w[N_{ng} + G_{ng}], \\ & w[S_{ng} : S_{ng} + N_{ng} - 1] \cdot w[S_{ng} + N_{ng} + G_{ng}], \\ & w[2S_{ng} : 2S_{ng} + N_{ng} - 1] \cdot w[2S_{ng} + N_{ng} + G_{ng}], \\ & \ldots, \\ & w[nS_{ng} : nS_{ng} + N_{ng} - 1] \cdot w[nS_{ng} + N_{ng} + G_{ng}]\rangle \quad : \\ & nS_{ng} + N_{ng} + G_{ng} = |w| - 1 \end{aligned}$$

The $w[i : j]$ is a part of the word w starting from position i (0-based index) to position j (inclusive), and $a \cdot b$ denotes concatenating operator of a and b symbol sequences.

Usually gapless situations occur and $G_{ng} = 0$, therefore in further discussions this parameter will be omitted.

There are several choices for transcription – raw ("shine" → "s", "h", "i", "n", "e"), phonetic ("shine" → "ʃ", "a", "ɪ", "n"), and for English language only semi-literal ("shine" → "sh", "i", "n", "e"). The raw transcription is the fastest and easiest to analyze, on the other hand phonetic transcription provides the best quality of result words, but they are hard to analyze (read as IPA, *International Phonetic Alphabet* codes) – for example "sɜːtnlɪ" (certainly). Mixing those two is the best choice (the system uses both or only phonetic form for computation, but results are presented in raw form). However, the constant conversion between those two forms requires a lot of time (conversion is done with external program, which has to be called for every conversion). Thus the third form was introduced, which is as fast as the raw one, but allow to keep most characteristic English phonemes as atomic symbols ("sh", "ch", "th", "ph", "wh", "rh").

Let us denote by $count(s, w)$ a function which returns the occurrence count of s symbol within w word, additionally let's denote the alphabet used by Σ. Some of available transformations can be written as:

null transformation	$w \to w$
extension	$w \to w{\cdot}w[0 : N_{ng} - 1]$
reflection	$w \to w[\lvert w\rvert - 1]{\cdot}w[\lvert w\rvert - 2]{\cdot}\ldots{\cdot}w[0]$
sorting	$w \to w'$: $\underset{0<i<\lvert w'\rvert}{\forall} w'[i-1] \leqslant w'[i]$, $\underset{a\in\Sigma}{\forall} count(a, w) = count(a, w')$

Examples of above transformations:

null transformation	"lollipop" → "lollipop"
extension	"lollipop" → "lollipoplo": for $N_{ng} = 2$
reflection	"lollipop" → "popillol"
sorting	"lollipop" → "illloopp"

As one can notice some transformations are reversible (e.g. extension), while others (like sorting) are not. It is possible to combine basic transformations into compound ones, as long as their parameters N_{ng} and S_{ng} are pair-wise equal. In order to get reversible compound transformation all its components have to be reversible.

Let us define three auxiliary functions:

$$\begin{aligned} h(w) &= w[0 : |w| - 2] && : |w| > 1 \\ t(w) &= w[|w| - 1] && : |w| > 1 \\ r(i) &= \begin{cases} i & \text{for positional n-grams} \\ 0 & \text{for unbounded n-grams} \end{cases} \end{aligned}$$

Parameter N_{ng} and function r control balance point in the creative process between factors of novelty and correspondence. Big value of N_{ng} is equivalent for case when after learning the system becomes passive and just repeats prior knowledge. Small values of N_{ng} and function r for unbounded n-grams work as completely spontaneous system.

The following functions and parameters describe the model of learning the words and creating them:

- function r — defines whether the result of ng is a sequence of n-grams or a multiset,
- n-grams parameters N_{ng}, S_{ng}, G_{ng}
- composition of transformations T,
- transcription Z,
- alphabet Σ,
- weights of matrices W.

Weight W of a matrix defines importance of given model used to estimate its elements. Since it is possible to use several models at once the overall word rank is product of all used models word ranks:

$$Q(w) = \prod_{k=0}^{\#models-1} Q'_k(w)^{W_k}$$

Only words with score above defined threshold are accepted as interesting – the threshold can change while the program is running.

At least one model has to be defined. When there are several models the first one is considered as the main one (builder), and the others are auxiliary (they do not participate in building words, only in scoring words). Because of those design choices the main model has to use only reversible transformations, and in case of using several different transcriptions, the main model has to use raw transcription.

Let g be the i-th n-gram in w word. It will be denoted as g_i. Let us define additionally:

$$\begin{aligned} P(b|a_i) =& P(b_{i+|a|}|a_i) \\ P(a_i) =& P(a|i) \end{aligned}$$

where $h(g) = a, t(g) = b$.

For short let us denote G as sequence of n-grams $G = ng(w)$. Then function $q(G)$ is defined as product of probabilities:

$$q(G) = P(h(G[0])_{r(0)}) \prod_{i=0}^{|G|-1} P(t(G[i])|h(G[i])_{r(i)})$$

or more generally, when position of G is arbitrary:

$$q(G, d) = P(h(G[0])_{r(d)}) \prod_{i=0}^{|G|-1} P(t(G[i])|h(G[i])_{r(d+i)})$$

where d is the n-gram starting index. The usefulness of this general formula is easy to notice when one considers an example: the system that learned word "heart" now can create the word "art" regardless of function r.

For main model and positional n-grams thanks to general formula for $q(G)$ it is possible to separate the value of n-gram and its index, and as the result sequence with gaps, split into independent sub-sequences $G_0, G_1, ..., G_J$. The final version of the formula for $q(G)$ is defined as follows:

$$q(G) = \max_{G_0,G_1,\ldots,G_J=G} \prod_{i=0}^{J} \max_{0 \leqslant d_i \leqslant depth(D)-|G_i|} q(G_i, d_i)$$

The parameter J is given by the system operator, it defines the number of sequence divisions; when $J = 0$ the sequence G is not divided into parts (in practice it is assumed that d_0 is equal 0). When $J > 0$ then for each part of G it is valid to set different model of computation.

It is possible to introduce penalty function for jumping, such that the penalty equal to 0 means forbidding jumping, and 1 means no penalty at all. The penalty function should be defined in such way that in the case of jumping between subsequent positions or in case of jumping to undefined position, its value is equal to 1. In other cases it is computed as:

$$pn(i,j) = \frac{\sum\limits_{b\in\Sigma, a\in\Sigma^{Nng}} bin(P(b|a_{r(i)}))bin(P(b|a_{r(j)}))}{\sum\limits_{b\in\Sigma, a\in\Sigma^{Nng}} bin(P(b|a_{r(i)}))}$$

where bin function is defined as follows:

$$bin(x) = \begin{cases} 0 & : x = 0 \\ 1 & : x \neq 0 \end{cases}$$

One can easily notice that in fact $pn(i, i+1) = 1$. $q(G)$ version with penalty is defined below:

$$\begin{aligned} q_{pn}(G) \quad &= \max_{G_0,G_1,\ldots,G_J=G} \\ \prod_{i=0}^{J} & \max_{0 \leqslant d_i \leqslant depth(D)-|G_i|} pn(d_i + |G_i|, d_{i+1}) q(G_i, d_i) \end{aligned}$$

Similarity

Words created by the algorithm should be original, so duplicates of previously learned words should be eliminated. At the similarity check module all abbreviations of known words are rejected (“borrow" $\rightarrow$ “borr"), and words with superficial changes (“borrow" $\rightarrow$ “borrom"). At first it looks like obvious decision, but it is not the case – if one had used the same principle for a real language it would be too prohibiting. It is enough to consider such words like “sorrow", “barrow", or “burrow". However originality is a crucial issue for the purpose of creative system: product names should be different than already existing ones (yet, sometimes similarity has advantages, for example there are several companies with names like “Imtel" or “Indel"). The same rules apply when checking words which are similar to previously created.

For computing similarity between words, edit distance algorithm is used (Damerau-Levenshtein version [32]), where the possible operations are:

- deletion, e.g. “earn" $\rightarrow$ “ear",
- insertion, e.g. “sting" $\rightarrow$ “string",
- exchange, e.g. “neat" $\rightarrow$ “heat",
- transposition of adjacent characters, e.g. “quiet" $\rightarrow$ “quite".

The cost for each of the above operations is set to 1. Difference between word and set of words is equal to:

$$\text{diff}(w, Dict) = \min_{v\in Dict} edist(v, w)$$

where $edist$ denotes edit distance function, w – a word being checked, and $Dict$ – set of words (the dictionary or already created words). The value of above function equal to 0 means that the word being tested and some word from the word set are identical. The word is accepted if the function value is greater than system threshold (set by the system operator). Usually the threshold for checking against dictionary was set to 2, and for already created words to 1. Checking for abbreviations is non-parametric, i.e. no matter how much the word is truncated it will always be rejected.

The possibility of using open-bigrams mechanism was also considered [33] for calculating similarity, but because of its slightly different purpose it was finally abandoned.

Associations

The purpose of this task is checking associations at the transcription level and detecting association hijacking. When one is asked whether the created word “tubowle" contains priming word “owl" the answer will be positive (“tub$\underline{\text{owl}}$e"). When one is asked generally what words are associated with this one, the answer will be rather “bowl" or “tub". In this example “bowl" dominated the word “owl" (more in [34]).

Separation of associations is a difficult task. At the semantic level this effect can be considered as useful one [35], but the solution explained below is the most unreliable part of the entire system. For each word from the priming set a prefix and a suffix indicator is computed. The prefix (suffix) indicator is the shortest prefix (suffix) of the given word, which is not a prefix (suffix) of any other word from the dictionary with the exception of inflections and derivations of that word. It is possible that there is no such prefix (suffix) and entire word has to be its indicator. For example the prefix indicator of the word “education" is “educat" – there are words that share this prefix, like “educated" and "educational", but they are treated as related to “education".

The purpose of the module is to find matches between created word and indicators from the priming set. In general it is the LCS (*longest common substring*) problem, here not in the "1:1", but in the "1:many" version, with no overlaps of the matches condition. The operator of the system sets such parameters:

1) uniqueness level of the matching,
2) the minimal number of symbols for a match for each indicator,
3) two matches cannot overlap with optional exception of one shared symbol,
4) whether matching with prefixes, infixes, suffixes is allowed in case of shortening of the indicators,
5) the mode of matching – whether it should be (left) greedy or non-greedy. In the first case the position of

the match is fixed first, and then as many symbols as possible from this position to the right symbols are taken.

Here is the brief explanation of the above parameters:

1) the system can work of any level of those three: indicator, group or concept. Assuming there is a concept (idea) consisting of two groups “walking", “walks" and “running", “runner", at the level of the indicator, any of those words can be matched, at the level of the group “walking" and “walks" cannot be matched at the same time and also "running" and "runner" cannot be matched at the same time (in other words – only one from the group can be used), and at the concept level only one word can be matched;
2) consider testing “pleasure" with indicators “please" and “sure". If requirement for the first indicator is to match all symbols, the matching will fail. However, if the requirement is four symbols there will be a match – "plea••••", where symbol "•" means "ignored" (not important);
3) using the same words as above; with limits of matches equal to 5 (“please") and 4 (“sure") the matching will fail, however if operator allows to share one symbol, the matching will succeed – for the first indicator there will be “pleas•••", and for the second “••••sure";
4) using the same words as above, with the requirement of suffix matching, the first indicator can match only in two ways for one symbol each, with the requirement of prefix matching single one for maximum of five symbols, and with requirement of infix matching eight matches ("pleas•••", "•leas•••", "••eas•••", "••e•••••", "•••••••e", "•••••••e", "•••as•••", "••••s•••" – the longest matches were shown);
5) using the same words as above, in the prefix mode, greedy, with no overlaps, no matter what are the limits, the matching will always fail – “please" matches before “sure" does, and thus it has priority in matching maximum number of symbols – “pleas•••" – for the second indicator there is no place to make a match (prefix one, see the assumption on the beginning of this point).

The way this module work focuses on attracting associations, however there is no inhibiting mechanism built in. It can lead to overflow of associations, and as effect slower perception by human ([36]) – in practice such word will be unwanted, but current version of the system will accept it.

IV. Results

Think about an e-book reader, such as Amazon Kindle. It is an electronic device, a kind of a computer, that is used for reading books, news, words, learning, education, acquiring, collecting, gathering information, data, discovering knowledge, building library, memory, natural, personal portable light device that is easy to use by humans, anyplace, anytime, anywhere, on the move, journey, going to travel the world, globe, using remote wireless networks, without cables, replacing paper books. This type of description is sufficient to provide priming of the system that is used to create names for e-book readers. In addition a few words that should exclude some associations may be added.

A large number of experiments with different settings of parameters described above has been performed (Pilichowski, PhD thesis). Many words created in this way by the BrainGene algorithm have already been used as names of some products, companies or web sites, indicating that the algorithm has captured the gist of the brain processes behind creative thinking in this simple domain. For example, *infoworld* is frequently used, *inforion* is not that common but is a name of a company (inforion.pl) that provides information services. Many other names are used in the .com addresses or as company names, not necessarily connected to e-books, for example: *bookist, boomation, bookstion, cablects, cablected, cablector, collead, dataction, datamation, datnews, datmation, easnews, educatics, electroad, explobal, goinmation, gonewsy, infordata, inforld, inforlds, inforion, infornews, infortion, infonews, inforvel, infravel, lighbooks, newsion, newstion, papnews, travelation, travelnews, wentnews*. Some internet domains are still for sale (February 2013), for example *cablead, easmation, educatnews, infovel, pocketnews, wortion* with prices exceeding 1000 USD.

Some interesting words created by the BrainGene algorithm are novel, for example *acquirave, bookists, cablect, datorld, explorlds, gonalness, infoion, inforldly, inforravel, journics, libravely, memotedly, memorld, memorlds, memovely, newravel, roadmation, wornews*. Some words are family names or nicknames, for example *Daturave, Jourary, Wentled*, but have not been used as product names.

There is also a service *http://domomark.com/* that evaluates how good are domain names, checking distinctivity of the names and assigning scores to them in quite naive way. BrainGene algorithm certainly provides much more sophisticated way of domain quality evaluation then offered by *domomark*, and all names given above have high marks when evaluated by this service.

V. Conclusions

Inventing novel names is clearly a creative process that can be simulated using computational algorithms, and tested against human ingenuity. Many companies offer naming services, for example: Brands and Tags, Brighter Naming, Names and Brands, Named at Last, NameSharks. Sometimes competitions for a good name are organized. Automatic invention and evaluation of usefulness of names is an interesting engineering problem. In the language domain competition of programs with people is always very difficult. Except for generation of names that may have commercial value the same approach can be used to generate passwords that are easy to remember but cannot be found in the dictionary.

General neurocognitive principles of creativity are probably common to various tasks [16], [5]. Models of neurocognitive processes involved in the process of creating and understanding novel words, capturing details at different levels of approximation of brain functions, are worth developing. In this paper a few psychological approaches to creativity have been discussed and some details related to the new developments of our probabilistic approach to creation of novel pseudo-words have been presented. Despite its simplicity the model gives quite interesting results and the access to the *BrainGene* server that implements the algorithm may be obtained from the authors.

Some novel names invented by the early version of this algorithm have already been published [5]. For example, the name "braingene" has been generated for the program itself based on description of its functionality. In various test runs about 2/3 of all the names proposed by the program has already been used as company or internet site names, but the rest is novel and of similar quality. This gives us confidence that the algorithm indeed approximates some processes behind creative thinking in this restricted domain.

Although the interaction between brain activities responsible for phonological and semantical representations is quite complex, and at this stage we have found it easier to use probabilistic model where some rules may be incorporated in a direct way, in principle neural models of this sort could be created, extending representation of the words by adding the meaning based on distribution of brain activities for a given word or concept. The meaning of some concepts has already been linked to the brain activity [37], and a sketch for the brain-based representation of concepts has been presented [19]. There is no doubt that algorithms based on neurocognitive inspirations are going to be quite useful in modeling creative processes, although the way towards human-level competence in this domain will be quite long.

REFERENCES

[1] J. Goldenberg and D. Mazursky, *Creativity in product innovation*. Cambridge University Press, 2002.

[2] J. M. Zytkow and H. A. Simon, "A theory of historical discovery: The construction of componential models," *Machine Learning*, vol. 1, p. 107, 1986.

[3] B. Goertzel, *From complexity to creativity*. Springer, 1997.

[4] S. Harnad, *Creativity: Method or Magic?* [Online]. Available: http://cogprints.org/1627/1/harnad.creativity.html

[5] W. Duch and M. Pilichowski, "Experiments with computational creativity," *Neural Information Processing - Letters and Reviews*, vol. 11(4-6), pp. 123–133, 2007.

[6] M. Pilichowski and W. Duch, "Neurocognitive approach to creativity in the domain of word-invention," *Lecture Notes in Computer Science*, vol. 5507, pp. 88–96, 2009.

[7] W. Duch, "What is computational intelligence and where is it going," in *Challenges for Computational Intelligence*, W. Duch and J. Mandziuk, Eds. Springer, 2007, vol. 63, pp. 1–13.

[8] J. Hadamard, *An Essay on The Psychology of Invention in the Mathematical Field*. Dover Publications, 1949.

[9] J. Goldenberg, D. Mazursky, and S. Solomon, "Templates in creative sparks," *Science*, vol. 285, no. 5433, 1999.

[10] G. Altshuller, *Algorithm of Invention*. Moscowskiy Rabochy, 1969.

[11] S. Savransky, *Engineering of Creativity*. CRC Press, 2000.

[12] A. Dijksterhuis and T. Meurs, "Where creativity resides: The generative power of unconscious thought," *Consciousness and Cognition*, vol. 15, no. 1, pp. 135–146, 2006.

[13] A. Dijksterhuis and L. F. Nordgren, "A theory of unconscious thought," *Perspectives on Psychological Science*, vol. 1, no. 2, pp. 95–109, 2006.

[14] M. Boden, "What is creativity?" in *Dimensions of Creativity*, M. Boden, Ed. The MIT Press, 1996, ch. 4, pp. 75–118.

[15] J. Koza, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 251–284, 2010.

[16] W. Duch, "Computational creativity," in *World Congres on Computational Intelligence, Vancouver, Canada*. IEEE Press, 2006, pp. 1162–1169.

[17] ——, "Intuition, insight, imagination and creativity." *IEEE Computational Intelligence Magazine*, vol. 2(3), pp. 40–52, 2008.

[18] W. Duch, P. Matykiewicz, and J. Pestian, "Towards understanding of natural language: Neurocognitive inspirations," *Lecture Notes in Computer Science*, vol. 4669, pp. 953—-962, 2007.

[19] ——, "Neurolinguistic approach to natural language processing with applications to medical text analysis." *Neural Networks*, vol. 21(10), pp. 1500–1510, 2008.

[20] D. Campbell, "Blind variation and selective retention in creative thought as in other knowledge processes," *Psychological Review*, vol. 67, p. 380–400, 1960.

[21] D. Simonton, "Creative thought as blind-variation and selective-retention: Combinatorial models of exceptional creativity," *Physics of Life Reviews*, vol. 7, p. 156–179, 2010.

[22] F. Pulvermüller, *The Neuroscience of Language. On Brain Circuits of Words and Serial Order*. Cambridge Uni. Press, 2003.

[23] F. P. M. Garagnani, T. Wennekers, "A neuroanatomically-grounded Hebbian learning model of attention-language interactions in the human brain," *European Journal of Neuroscience*, vol. 27(2), pp. 492–513, 2008.

[24] F. Pulvermüller, "Meaning and the brain: The neurosemantics of referential, interactive, and combinatorial knowledge," *Journal of Neurolinguistics*, vol. 25, p. 423–459, 2012.

[25] L. R. Iyer, S. Doboli, A. A. Minai, V. R. Brown, D. S. Levine, and P. B. Paulus, "Neural dynamics of idea generation and the effects of priming," *Neural Networks*, no. 2009 Special Issue, 2009. [Online]. Available: www.elsevier.com/locate/neunet

[26] G. A.Miller, C. Fellbaum, R. Tengi, P. Wakefield, H. Langone, and B. R. Haskell, *A lexical database for the English language*. [Online]. Available: http://wordnet.princeton.edu/

[27] T. Brants and A. Franz, *Web 1T 5-gram Version 1*. [Online]. Available: http://www.ldc.upenn.edu/Catalog/

[28] *The Specialist Lexicon*. [Online]. Available: http://lexsrv3.nlm.nih.gov/LexSysGroup/Projects/lexicon/

[29] K. Atkinson, *Spell Checker Oriented Word Lists*. [Online]. Available: http://wordlist.sourceforge.net/

[30] M. Spitzer, *The Mind within the Net: Models of Learning, Thinking, and Acting*. MIT Press, 2000.

[31] W. Duch, "Intuition, insight, imagination and creativity," *IEEE Computational Intelligence Magazine*, vol. 2, no. 3, pp. 40–52, 2007.

[32] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[33] J. Grainger and C. Whitney, "Does the huamn mnid raed wrods as a wlohe?" *Trends in Cognitive Sciences*, pp. 58–59, 2004.

[34] J. S. Bowers, C. J. Davis, and D. A. Hanley, "Automatic semantic activation of embedded words: Is there a "hat" in "that"?" *Journal of Memory and Language*, vol. 52, pp. 131–143, 2005.

[35] W. Duch and K. Dobosz, "Visualization for understanding of neurodynamical systems," *Cognitive Neurodynamics*, vol. 5, pp. 145–160, 2011.

[36] M. S.Vitevitch, "The spread of the phonological neighborhood influences spoken word recognition," *Memory & Cognition*, vol. 35, no. 1, pp. 166–175, 2007.

[37] T. Mitchell, S. Shinkareva, A. Carlson, K. Chang, V. Malave, R. Mason, and M. Just, "Predicting human brain activity associated with the meanings of nouns," *Science*, vol. 320, no. 5880, pp. 1191–1195, 2008.