Annales Univ. Sci. Budapest., Sect. Comp. 36 (2012) 299–322

AUTOMATED WORD PUZZLE GENERATION USING TOPIC MODELS AND SEMANTIC RELATEDNESS MEASURES

Balázs Pintér, Gyula Vörös,
Zoltán Szabó and András Lőrincz
(Budapest, Hungary)

Communicated by András Benczúr

(Received December 21, 2011; revised February 9, 2012;
accepted February 14, 2012)

Abstract. We propose a knowledge-lean method to generate word puzzles from unstructured and unannotated document collections. The presented method is capable of generating three types of puzzles: odd one out, choose the related word, and separate the topics. The difficulty of the puzzles can be adjusted. The algorithm is based on topic models, semantic similarity, and network capacity. Puzzles of two difficulty levels are generated: beginner and intermediate. Beginner puzzles could be suitable for, e.g., beginner language learners. Intermediate puzzles require more, often specific knowledge to solve. Domain-specific puzzles are generated from a corpus of NIPS proceedings. The presented method is capable of helping puzzle designers compile a collection of word puzzles in a semi-automated manner. In this setting, the method is utilized to produce a great number of puzzles. Puzzle designers can choose and maybe modify the ones they want to include in the collection.

Key words and phrases: Natural language processing, puzzle generation, word puzzles, topic model, semantic relatedness, Wikipedia.
1998 CR Categories and Descriptors: I.2.7.
The Research is supported by the European Union and co-financed by the European Social Fund (grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003).

1. Introduction

Puzzles are frequently used as assessments in education and psychometry [29]. For example, odd one out puzzles are often found in IQ tests (e.g., in [6]). Word puzzles are often designed to test or improve a wide array of skills, for example, language skills, verbal aptitude, logical thinking or general intelligence. Multiple-choice synonym questions are part of the TOEFL test[1].

To solve these problems, one needs to recognize which words are related. Words can be related in different ways. Consider the following odd one out puzzle: *salmon, shark, whale, elephant*. Here the odd one out is *elephant* because all the others live in water, which is a common attribute of the concepts the words denote. In the following problem: *table, level, racecar, civic*, the first word is the odd one out because all the other words are palindromes, which is an attribute of the word forms, not the concepts. In this puzzle: *battle, army, attack, book*, the last one is the odd one out because the others are all connected to a single topic, *warfare*.

Designing and maintaining a collection of word puzzles is time-consuming. A large amount of material is required to maintain variety; otherwise the solver will encounter the same puzzle multiple times. Moreover, languages are constantly changing: new words are created (e.g., *blog*), existing words get new meaning (e.g., *chat*), words go out of common use (e.g., *videotape*). New puzzles are needed all the time to keep up to date, or to test new knowledge. Automated generation of word puzzles would be of considerable benefit.

In this paper, we present a method that automatically generates word puzzles that can be solved by differentiating the words in the puzzle based on their *topic*. The method is unsupervised, only a corpus of unlabeled documents is needed as input[2]. Generating domain-specific word puzzles is possible by using domain-specific corpora. The method is also language-independent: the only language-dependent components are the stemmer and the stopword list.

We consider the following classes of word puzzles: *odd one out*, *choose the related word* and *separate the topics*. In each puzzle, a small number of words are presented to the solver, who is then required to select some of them according to the instructions of the puzzle class:

- In *odd one out* puzzles, the solver is required to select the word that is dissimilar to the other words.

[1] Test of English as a Foreign Language, `http://www.ets.org/`

[2] To measure semantic relatedness, we use Explicit Semantic Analysis that requires Wikipedia as background knowledge. However, other semantic relatedness measures could be used.

- In *choose the related word* puzzles, the solver has to select the word that is closely related to a previously specified group of words.
- In *separate the topics* puzzles, the solver has to separate the set of words into two disjoint sets of related words.

To work out a general method, it is helpful to observe that all three puzzles require sets of words with similar meaning. For example, in *odd one out*, there is a set of similar words, and a single dissimilar word (i.e., the odd one).

Sets of similar words can be obtained by employing *topic models*. In natural language processing, documents are often modeled as combinations of latent topics. The topics are determined directly from the corpus of documents. A topic is essentially a set of words that are similar in meaning, so methods that learn topics of documents can also generate these sets.

However, these algorithms also produce many, so-called *junk topics* that do not pass as sets of similar words [2]. For example, common function words, such as *did, said*, etc. can form a topic, which cannot be interpreted as a *consistent set* of related concepts. These topics have to be identified and discarded.

To form a *consistent set*, the set of the most significant words in the topic is considered. This set is *consistent* if there are no unrelated words in it. Semantic relatedness between pairs of words is determined by Explicit Semantic Analysis [13] (Section 4).

To diminish the errors made by the semantic relatedness measure, we consider words as nodes of a *network*, where the weight of the edges are determined by the relatedness measure. The capacities in this network are examined in order to determine whether a set is consistent (Section 5.1). Puzzles are generated by adding dissimilar items (e.g., words or other sets) to consistent sets.

In the next section, we briefly introduce automatic puzzle generation. The presented method is built upon two pillars: topic models, and semantic relatedness measures, presented in Section 3 and Section 4. After the method (Section 5), the results are discussed (Section 6), and we conclude in Section 7.

2. Automatic puzzle generation

Procedural content generation for games (PCG) is the automated generation of game content, such as maps, textures or puzzles. The popularity of video games is increasing: the US Entertainment Software Association reports that as of 2011, 72% of American households play computer and video games[3] in

[3] `http://www.theesa.com/facts/pdfs/ESA_EF_2011.pdf`

contrast to 67%, reported in 2010. With the increasing popularity of video games, there is a growing demand for game content. However, generating content manually is expensive. Therefore, PCG is more and more important, as shown by, for example, the second *International Workshop on Procedural Content Generation in Games* [1].

Generating puzzles is a subfield of PCG. As puzzles can be very different from each other, the approaches used for generating puzzles also vary considerably. Generating and solving sudoku puzzles are particularly popular, and many successful algorithms have been proposed for them, such as constraint programming [25] or graph transformation methods [12]. Genetic algorithms are also utilized to create various puzzles, such as the mazes on chessboards generated by Ashlock [3]. There is great interest in generating puzzles and quests (objectives for the players) for Massively Multiplayer Online Games [10, 18].

The area of automated *word puzzle* generation is – to the best of our knowledge – a field that has not yet been studied extensively. Colton [8] used a complex theory formation system to generate *odd one out*, *analogy* and *next in the sequence* puzzles. In contrast to his method, the method presented in this paper does not require highly structured datasets to generate the puzzles.

3. Topic models

Topic models are based on the assumption that documents can be described as mixtures of some latent (i.e., unobserved) topics. For example, a text about teaching algebra may be described by as the mixture of the following topics, among others: *education*, *mathematics*. Another example is shown in Figure 1.

Both the topics and documents are usually taken to satisfy the *bag of words* assumption: the order of the words is not considered important, and is discarded. A topic is usually modelled as a set of weighted words, each word having a weight according to its relevance to the topic. For example, the words of the topic about *mathematics* may be the following: (*mathematics*, 0.9), (*math*, 0.7), (*calculus*, 0.3), (*algebra*, 0.3), etc.

The documents of a corpus are represented as *term vectors*. Each document is represented as a vector $\mathbf{x}$ of weights assigned to words, where a weight x_i is the number of occurrences of the ith word in the document (i.e., the *term frequency*). This vector $\mathbf{x}$ is called the *term vector* of the document.

From these representations, a term-document matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M] \in \in \mathbb{R}^{N \times M}$ can be constructed, where each of the M columns is the N-dimensional term vector of a document.

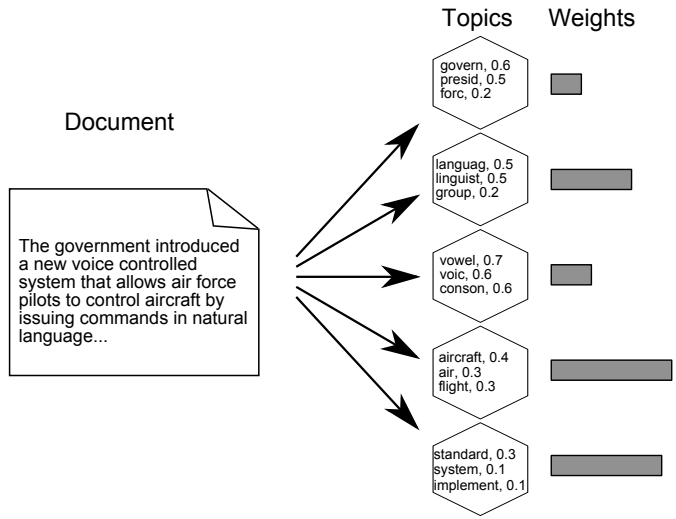


Figure 1. Topic modeling. In this example, the document is represented as the combination of five topics. The weights in the representation $\boldsymbol{\alpha}_i$ describe the extent to which each topic is characteristic of the document

An alternate representation of documents can be given by transforming the term-document matrix into a topic-document matrix, where each column contains the weights of topics for a document.

In this paper, we consider three different topic models. *Latent Semantic Analysis* (LSA) [9] and *Online Group-Structured Dictionary Learning* (OSDL) [28] both factorize the term-document matrix $\mathbf{X}$, while *Latent Dirichlet Allocation* (LDA) [4] is a generative probabilistic model.

3.1. Latent Semantic Analysis

Latent Semantic Analysis (LSA) [9] is perhaps the most widely known topic model. LSA uses *singular value decomposition* (SVD) to capture the hidden correlations between words. SVD is used to factorize the term-document matrix into the product of three special matrices: $\mathbf{X} = \mathbf{USV}^T$. The columns of $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and are called the left and right *singular vectors* of $\mathbf{X}$. $\mathbf{S}$ is a diagonal matrix, whose entries are called the *singular values* of $\mathbf{X}$. The singular values are non-negative and are sorted in descending order.

An approximation of $\mathbf{X}$ can be obtained by keeping only the K largest singular values, and deleting the other rows and columns of $\mathbf{S}$ (along with the corresponding columns of $\mathbf{U}$ and $\mathbf{V}$). Let $\hat{\mathbf{U}}$, $\hat{\mathbf{S}}$ and $\hat{\mathbf{V}}$ denote the matrices derived from $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$ by keeping only the K largest singular values (Fig-

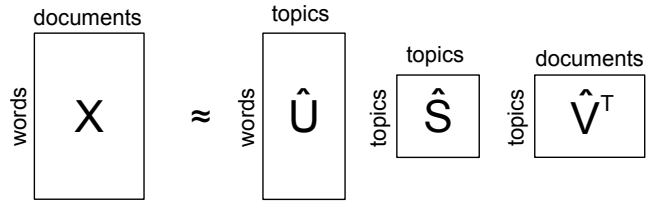


Figure 2. Partial singular value decomposition to get a low-rank approximation of the term-document matrix $\mathbf{X} \approx \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T$

ure 2). The resulting factorization is called *partial* SVD, and it generates an optimal K-rank approximation of $\mathbf{X}$:

$$\underset{\mathrm{rank}(\hat{\mathbf{X}})=d}{\arg\min} \left|\left|\mathbf{X} - \hat{\mathbf{X}}\right|\right|_F = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T. \tag{3.1}$$

Equation 3.1 can be interpreted as approximating columns of $\mathbf{X}$ (i.e., the documents) as combinations of K latent topics, held in the left singular vectors of $\mathbf{X}$. A right singular vector contains the weights of the topics for a document.

Latent semantic analysis can be successfully applied in many areas of natural language processing, such as text segmentation [7]. The application of LSA in information retrieval is called *latent semantic indexing*.

3.2. Online Group-Structured Dictionary Learning

Online Group-Structured Dictionary Learning [28] is a recent algorithm that factorizes the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$ into two matrices: the *dictionary* $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K] \in \mathbb{R}^{N \times K}$ and the *representation* $\mathbf{A} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_M] \in \mathbb{R}^{K \times M}$. The dictionary $\mathbf{D}$ contains the topics as columns. There are K topics, where K, the dimension of the vectors $\boldsymbol{\alpha}_i$, is a parameter.

Each document $\mathbf{x}_i$ is represented as a linear combination of topics. $\mathbf{A}$ contains the coefficients of these linear combinations: $\mathbf{x}_1 \approx \mathbf{D}\boldsymbol{\alpha}_1$, $\mathbf{x}_2 \approx \mathbf{D}\boldsymbol{\alpha}_2$, etc.. In matrix notation: $\mathbf{X} \approx \mathbf{DA}$.

Two additional constraints are introduced in contrast to LSA. The document is described by the combination of *a few groups of topics*, and the topics are embedded into a *topography*, where topics that are near to each other are more related than distant topics (Figure 3).

In contrast to LSA, this algorithm is *online*. The $\mathbf{x}_i$ are processed in sequence, and the dictionary $\mathbf{D}$ is updated in each iteration.

In the topography, a topic and its neighbouring topics constitute a group G_i. The groups form a family of sets $\mathcal{G} = \{G_i\} \subseteq 2^{\{1,\ldots,K\}}$, where each G_i

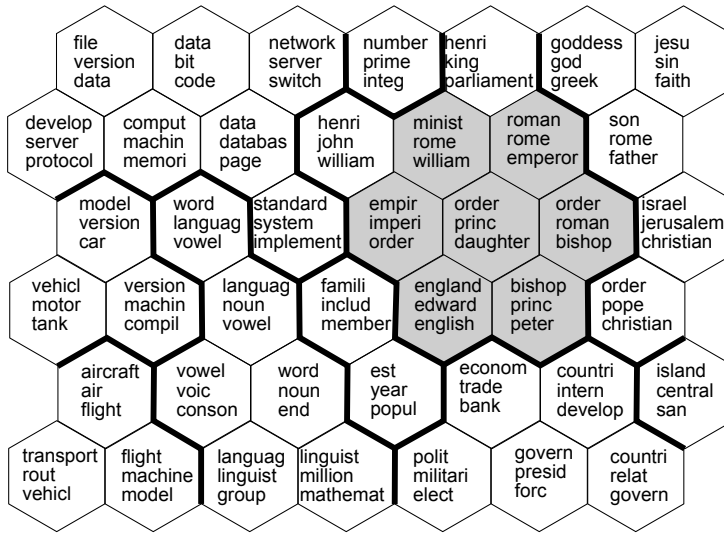


Figure 3. Part of a topography of topics produced by OSDL. The topics are embedded in a hexagonal grid on a torus. The topics were generated automatically. A single group is shown in grey. Clusters of similar topics are separated manually by wider lines in the figure. The words are stemmed

contains the indices of the topics in the group. For example, a family of groups $G_i = \{i\}$ would mean that each topic has only one neighbour, itself.

In this paper, the topography is a hexagonal grid on a torus. Each topic has exactly six neighbours, so $|G_i| = 7$ (every topic is also its own neighbour).

The additional constraints are realized as a structured sparsity inducing regularization. Each document is represented by a small number of groups G_i. In other words, the representation $\boldsymbol{\alpha}_i$ should be *sparse* with respect to the groups. The representation in each group can be dense, that is, multiple topics can be selected from within a group. Intuitively, each group can be thought of as a domain that contains related topics. The end result is that related topics are close to each other in the topography.

Formally, the OSDL problem is defined as the minimization of the following cost function:

$$\min_{\mathbf{D},\{\boldsymbol{\alpha}_i\}_{i=1}^{M}} \frac{1}{\sum_{j=1}^{M}(j/M)^{\rho}} \sum_{i=1}^{M} \left(\frac{i}{M}\right)^{\rho} \left[\frac{1}{2}\left\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\right\|_2^2 + \kappa\Omega(\boldsymbol{\alpha}_i)\right] (\kappa > 0), \tag{3.2}$$

$$\Omega(\boldsymbol{\alpha}) = \left(\sum_j \left\|\boldsymbol{\alpha}_{G_j}\right\|_2^{\eta}\right)^{\frac{1}{\eta}}, \tag{3.3}$$

where $\boldsymbol{\alpha}_{G_j} \in \mathbb{R}^{|G_j|}$ denotes the vector where only the coordinates that are in the set $G_j \subseteq \{1, \ldots, K\}$ are retained.

The first term is the approximation error, the second is the structured sparsity inducing regularization. For the case of $\rho = 0$, (3.2) is reduced to the empirical average, where every example $\{\mathbf{x}_i\}_{i=1}^M$ is present with the same weight $(\frac{1}{M})$ in the optimalization of $\mathbf{D}$:

$$\min_{\mathbf{D}, \{\boldsymbol{\alpha}_i\}_{i=1}^M} \frac{1}{M} \sum_{i=1}^{M} \left[\frac{1}{2} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \kappa \Omega(\boldsymbol{\alpha}_i) \right]. \tag{3.4}$$

The parameter ρ can be interpreted as a *forgetting rate*. Increasing ρ realizes the exponential forgetting of $\mathbf{x}_i$, with greater effect for larger ρ. In practice, employing forgetting can result in faster optimalization.

The parameter κ controls the tradeoff between the approximation error and the structured sparsity inducing regularization. The parameter η can be set to $0 < \eta \leq 1$. A smaller η results in stronger sparsification.

3.3. Latent Dirichlet Allocation

Probabilistic topic models [4,27] specify a probabilistic procedure by which documents are *generated*. The model of a corpus is obtained by inverting this process: the set of topics that could have generated the corpus is *inferred* by statistical techniques.

In *probabilistic latent semantic analysis* (PLSA) [16], each word is drawn from a *distinct topic*. For each document, a different distribution over topics is chosen. Each word in a document is generated independently by selecting a topic according to the per-document topic distribution, and drawing the word from that topic. Therefore, the joint probability model is

$$P(d_i, w_j) = P(d_i) \sum_{k=1}^{K} P(w_j|z_k) P(z_k|d_i), \tag{3.5}$$

where d_i is the label of the document generated, w_j are the words, z_k are the topics, and K is the number of topics.

PLSA is not a well-defined generative model [4]. The document label d is a dummy index into the list of documents in the *training set*. There is no natural way to assign probability to a previously unseen document. Another, more practical consequence is that the number of parameters to estimate grows linearly with the number of documents in the training set.

To overcome these problems, *latent Dirichlet allocation* (LDA) [4] (Figure 4) treats the per-document topic distributions θ_j, $(j = 1, \ldots, M)$ as a

K-parameter *hidden random variable*, where K is the number of topics. The *Dirichlet* distribution is chosen with parameter α, as it is conjugate prior to the multinomial distribution (i.e., the topic distribution of the documents).

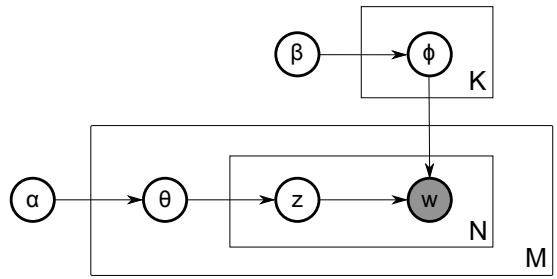


Figure 4. Smoothed latent Dirichlet allocation, plate notation. There are K different topics, M documents and N words

A problem of probabilistic topic models is that they assign zero probability to words that did not appear in the training corpus. To cope with this problem, usually smoothing is employed: positive probability is assigned to every word, whether or not they appeared in the training set. LDA performs smoothing by treating the mixture components ϕ_i, $(i = 1, \ldots, K)$ as a V-parameter Dirichlet hidden random variable with parameter β, where V is the size of the vocabulary.

The generative process LDA assumes for the whole corpus is the following.

1. Choose $\phi_i \sim \mathrm{Dir}(\beta)$, where $i \in \{1, \ldots, K\}$.
2. Choose $\theta_j \sim \mathrm{Dir}(\alpha)$, where $j \in \{1, \ldots, M\}$.
3. For each of the words $w_{j,t}$, where $t \in \{1, \ldots, N_j\}$
   (a) Choose a topic $z_{j,t} \sim \mathrm{Multinomial}(\theta_j)$.
   (b) Choose a word $w_{j,t} \sim \mathrm{Multinomial}(\phi_{z_{j,t}})$.

Here, $w_{j,t}$ is the tth word of the jth document, N_j is the number of words in it, $z_{j,t}$ is its topic, and θ_j is its topic distribution. The word distribution for topic i is denoted by ϕ_i. The parameters of the Dirichlet priors are α and β.

The total probability of the model is

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\phi}|\alpha, \beta) = \prod_{i=1}^{K} P(\phi_i|\beta) \prod_{j=1}^{M} P(\theta_j|\alpha) \prod_{t=1}^{N_j} P(z_{j,t}|\theta_j) P(w_{j,t}|\phi_{z_{j,t}}), \tag{3.6}$$

where $\boldsymbol{W} = \{w_{j,t}\}$ is the corpus, $\boldsymbol{Z} = \{z_{j,t}\}$ are the topics, $\boldsymbol{\theta} = \{\theta_j\}$ are the per-document topic distributions, and $\boldsymbol{\phi} = \{\phi_{z_{j,t}}\}$ are the per-topic word distributions.

4. Semantic relatedness

The notion of *relatedness* is utilized in many natural language processing tasks (e.g., document classification, information retrieval, etc.). For example, the aim of document clustering is to group texts in such a way that texts in the same group are more related to each other, but there are many other applications, such as grading of short answers [21], electronic career guidance [15] and text classification [5]. Semantic relatedness can be defined on different levels, e.g., between words, sentences, longer text fragments, or whole documents.

A common method to measure the semantic relatedness of two texts is to generate term vectors from the documents, and use a vector similarity measure. Some of these similarity measures are reviewed in [30], one of the most common being the *cosine similarity* of the two vectors:

$$sim(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{||\mathbf{a}||_2 ||\mathbf{b}||_2}.$$

Relatedness of words can also be determined, usually with the help of external knowledge sources.

Explicit Semantic Analysis (ESA) [13, 14] is a method to represent words as vectors in a high-dimensional concept space. Semantic relatedness of words can be computed by similarity measures on these vectors. ESA uses a special corpus in which each document describes a certain concept. The authors of ESA used the articles of the English Wikipedia as a concept repository because it is comprehensive and constantly maintained.

The basic assumption of ESA is that if a word appears frequently in a Wikipedia article, then that article (seen as a concept) represents the meaning of the word to some degree. The method constructs a matrix $\mathbf{C}$, whose columns are the term vectors of the articles in Wikipedia: the weight in c_{ij} is the tf–idf score of word i in article j:

$$\mathrm{tfidf}(i, j) = \mathrm{tf}(i, j)\mathrm{idf}(i),$$

where tf is the term frequency (the frequency of the word in the document) and idf is the inverse document frequency (representing the importance of the word based on the number of documents it appears in), defined as

$$\mathrm{idf}(i) = \log \frac{|D|}{\mathrm{df}(i)},$$

where $|D|$ is the number of documents, and $\mathrm{df}(i)$ is the number of documents word i appears in.

The assumption is that words that appear in less documents are more informative. The ith row of this matrix is the *concept vector* of word i.

Consider the word *tree*. In the concept vector assigned to it, there are many concepts with high weights, such as the concepts Oak and Tree (graph theory), because the word *tree* appears frequently in these articles. It is easy to see how concept vectors can be used to measure semantic relatedness: for example, in the concept vector of the word *vertex*, the weight for Tree (graph theory) is also high, therefore, the cosine similarity of the two concept vectors (for the words *tree* and *vertex*) will also be relatively high.

ESA has been applied in many fields, such as information retrieval [11]. There are also multiple extensions to the standard ESA model, such as CL-ESA (a cross-language version) [26], and TSA (an extension which takes the temporal correlation of words into account) [23].

5. The method

In this section, the method of generating word puzzles is described. The only input needed is a *corpus* of *unlabeled documents*. This corpus is modeled as a combination of latent *topics*. Among these topics, *consistent sets* are identified. The puzzles are generated by mixing these sets with words or other sets that are not related.

The algorithm works in two phases. First, the consistent sets are generated. In the second step, the word puzzles are created.

5.1. Generating consistent sets

As a first step towards generating the consistent sets, a topic model (Section 3) is produced from the corpus. Only the resulting topics are used, all other information (e.g., the topic distribution of each document) is discarded.

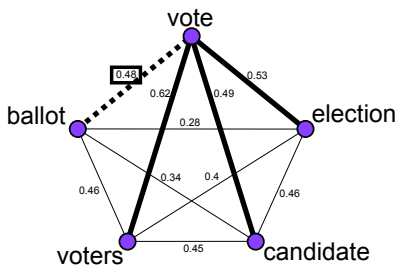
Sets of words are obtained by taking the k most significant words (i.e., those with the largest weights) of each topic. The sets are further examined to determine whether the words in the set are related. Only the sets with related words are kept for puzzle generation; these are referred to as *consistent sets*.

Even a single word too weakly connected to the others can make the resulting puzzles ambiguous. Therefore, each set must be rated according to the word that is the least related to the other words in the set.

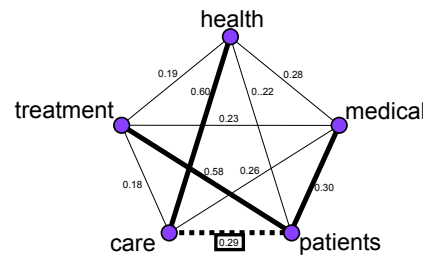
To measure the relatedness of two words, the cosine similarity of the concept vectors assigned by Explicit Semantic Analysis is used (Section 4).

Similarity measures are not perfectly accurate. They can make two types of errors. A *false positive* means that, according to the similarity measure, two words are related when in reality they are not. False positives may make us accept a set of unrelated words as related.

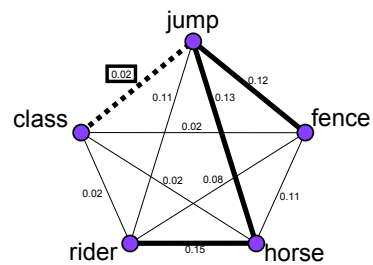
False negatives occur when the similarity measure gives a small value even though the two words are related. Because of false negatives, requiring that all pairs of words should be related is not a good criteria to determine whether a set of words is consistent.



(a) A highly consistent set.



(b) A consistent set.



(c) An inconsistent set.

Figure 5. Checking the consistency of three sets of words. In this example, each set contains five words. The edges of the maximal spanning trees are boldened. The edge with the minimal weight in the maximal spanning tree is denoted by a dashed line. The consistency of the set is defined as the weight of this edge. Figure 5(a) shows a very consistent set, where all the words are strongly connected to the word *vote*. The set on Figure 5(b) is consistent, but some of the relatedness values (e.g., between *care* and *treatment*) are lower than one would expect. The method is robust: a relatively high consistency value is assigned to the set. Figure 5(c) shows an inconsistent topic: the word *class* is weakly connected to the others

To deal with these errors, we determine the consistency of a set of words by constructing a *network*. A complete graph $G = (V, E)$ is constructed where

each node $v \in V$ corresponds to a word in the set. A mapping $w : E \to \mathbb{R}$ is defined on G, where $w(e)$ is the similarity computed by ESA between the endpoints (i.e., the words) of $e \in E$. The pair (G, w) is called a network [19].

We consider all the paths in G. It is assumed that if word v_1 is similar in meaning to word v_2, and word v_2 is similar to word v_3, then word v_1 is similar to word v_3. Even though this assumption does not hold in all cases, it can be utilized to make the procedure more robust to *false negatives*. Similarity between two words can be considered based on all the paths between them, additionally to the edges.

The method can be made more robust to *false positives* by defining the similarity of two nodes v_0, v_n given a path $W = v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ as the *capacity* of W,

$$c(W) = \min\{w(e_i) : i = 1, \ldots, n\}. \tag{5.1}$$

As the minimal edge weight is selected, values of the similarity measure larger than the real similarities have less effect on the result. To cope also with *false negatives*, the similarity between words $u, v \in V$ is defined as the capacity of a path of maximal capacity between them.

It seems that the capacity of all the paths between two nodes $u, v \in V$ need to be computed in order to determine their relatedness. Fortunately, we can use the following theorem [17, 19].

Theorem 5.1. *Let (G, w) be a network on a connected graph G, and let T be a maximal spanning tree for G. Then, for each pair (u, v) of vertices, the unique path from u to v in T is a path of maximal capacity in G.*

Based on this theorem, we determine the quality of a set as the minimum of the edge weights in the maximal spanning tree[4] of the graph G constructed from the set (Figure 5). In other words, the quality of the set is the similarity of the two most dissimilar words in the set. A set is consistent if its quality is above a predetermined threshold δ.

5.2. Creating the puzzles

Each word puzzle is generated by mixing unrelated elements with consistent sets. In *odd one out*, a single unrelated word is added to the set. In *choose the related word*, more unrelated words are added to a slightly larger set, where the unrelated words are the wrong answers. In *separate the topics*, two unrelated consistent sets are mixed.

[4]Maximal spanning trees are determined by *Kruskal's algorithm* [20].

As a first step, a corpus of documents is modeled by a topic model, and a consistent set of words is generated from each suitable topic as described in the previous section. The result is a family of consistent sets $\mathcal{T}$.

Semantic relatedness between two words is determined by the cosine similarity of their concept vectors produced by ESA. For words w and w', let $sim(w, w')$ denote their semantic relatedness.

In all three algorithms, a threshold θ determines whether the consistent set and the unrelated element are dissimilar enough so that they can be mixed to form a word puzzle. Another threshold, ϕ allows the creation of intermediate level *odd one out* and *choose the related word* puzzles (Section 6.3.2). By increasing ϕ, the relatedness of the additional elements (e.g., the *odd one out* word) to the consistent set is increased, therefore, the puzzle is made harder.

Algorithm 1 (Odd one out puzzle generation)

for all $T \in \mathcal{T}$ **do**
  repeat
    select random word w
    $\sigma \leftarrow \max_{t \in T} sim(t, w)$
  until $\phi < \sigma < \theta$
  output (T, w) puzzle
end for

An *odd one out* puzzle (Algorithm 1) consists of a set of related words, and another word which is not related to the ones in the set. The task of the solver is to find the word that is not related.

Algorithm 2 (Choose the related word puzzle generation)

for all $T \in \mathcal{T}$ **do**
  for $i = 1 \rightarrow k$ **do**
    $W = \emptyset$
    repeat
      select random word w
      $\sigma \leftarrow \max_{t \in T} sim(t, w)$
    until $\phi < \sigma < \theta$
    $W \leftarrow W \cup \{w\}$
  end for
  output (T, W) puzzle
end for

For *choose the related word* puzzles (Algorithm 2), two sets of words are generated: a consistent set, and k other words that are not related to any word in the set. The words are presented to the solver in a different grouping: one

word is moved from the consistent set to the set of unrelated words. So, in the latter set, one word is related to the words of the former. The task of the solver is to find that word.

Algorithm 3 (Separate the topics puzzle generation)

1: **for all** $T \in \mathcal{T}$ **do**
2: **repeat**
3: select random $T' \in \mathcal{T}$
4: $\sigma = \max_{t \in T, t' \in T'} sim(t, t')$
5: **until** $\sigma < \theta$
6: output (T, T') puzzle
7: **end for**

Separate the topics puzzles (Algorithm 3) consist of two sets of words where each word is related to all the other words within the same set, but is not related to any word in the other set. The task of the solver is to sort out the two sets.

5.3. Practical considerations

In addition to the abstract algorithms, there are some practical aspects of the puzzle generation process that must be considered.

An important consideration is how to choose the set the random unrelated word w (e.g., in the odd one out puzzle, the word that does not belong) is selected from. At first, we chose the set that contains every word in the corpus. However, we found that this set is too broad: there are many rare "words" (e.g., in Wikipeda: *erev*, *ern*) that should not be used as part of a puzzle. Thus, the set the unrelated words are chosen from must be chosen carefully. We opted for a straightforward solution: we use the union of all the words that appear in a consistent set of words.

Stemming (i.e., reducing inflected words to their stem) is important. Without stemming, puzzles such as *number, numbers, numbered, numberer, cat* are possible. Stemming reduces each of the four inflected forms of to their common stem, *number*, forcing the words in the puzzle to be different. Stemming should be carried out at the beginning of the puzzle generation process, before applying the topic models. In the final puzzles, the words are "unstemmed": they are replaced by their most common inflected form.

There is a possibility that no unrelated elements can be selected to form a puzzle with a consistent set. To prevent the algorithm running indefinitely, we only try to generate a puzzle from a consistent set a finite number of times. If the process is unsuccessful within that time, the set is discarded.

6. Results and discussion

The starting point of the presented method is a corpus of unstructured documents. We demonstrate the method on two corpora: a collection of full papers from the Neural Information Processing Systems (NIPS) conference [24], and a corpus obtained by randomly sampling the articles of Wikipedia.

The corpus compilation process is detailed in Section 6.1. Next (Section 6.2), we compare the topics generated by the three topic models described in Section 3. The consistency of the sets obtained from the topics generated by the three algorithms is compared.

The word puzzles generated by the presented method are examined in Section 6.3. First, the three kind of puzzles generated from the corpus of Wikipedia articles are examined. Then, the method is applied to produce domain-specific puzzles from the corpus of NIPS proceedings.

6.1. Collecting corpora from Wikipedia

The corpora of Wikipedia articles is generated as follows. We process an XML dump of the English Wikipedia downloaded from `http://dumps.wikimedia.org/enwiki/`. The words are stemmed using the *Porter* stemming algorithm [22]. A list of 571 stopwords are used to discard common function words such as *the, is, or*.

Articles less than 1000 non-stopwords long or have less than 20 incoming and 20 outgoing links were discarded. This step eliminates superfluous articles.

The term-document matrices $\mathbf{X}$ are compiled from Wikipedia as follows:

1. 50,000 articles are randomly selected.
2. The corpus is divided into 5 corpora, each with $M = 10,000$ articles.
3. A term-document matrix is created from each corpus.
4. Words occurring less than 100 times in the selected corpora are discarded.

6.2. Generating the consistent sets

Consistent sets are a cornerstone of the presented method. Each puzzle is generated by adding elements to such a set. In this section, we compare the number of sets of a given quality the different topic models can produce.

The quality of the sets can be adjusted by tuning the threshold δ. We model the two corpora (NIPS, Wikipedia) with each of the three topic models, and count the number of consistent sets produced from the topics for different values of the threshold. The consistent sets are composed of $k = 4$ words. We chose the number of topics to be $K = 400$.

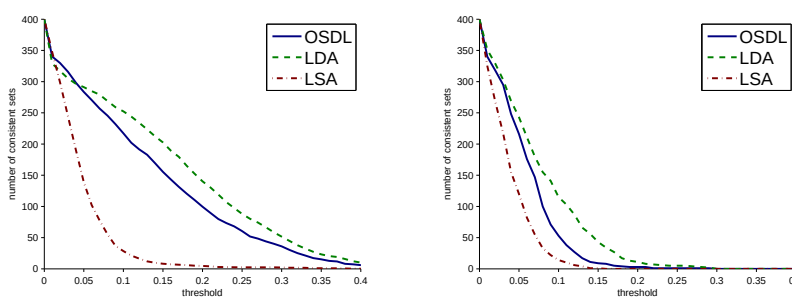
The topic models were applied as follows. Latent Semantic Analysis does not require any parameters (apart from the number of topics). For Latent Dirichlet Allocation, we set the parameters as suggested by [27]: $\alpha = 50/K$, where T is the number of topics, and $\beta = 0.01$.

For OSDL, the parameters were set as follows: $\eta = 0.5$ for strong sparsification, the learning rate $\rho = 1$, $\kappa = 2^{-13}$ for the corpus of Wikipedia articles, and $\kappa = 2^{-14}$ for the corpus of NIPS proceedings. The values of κ were determined experimentally.

Figure 6 shows the results. Figure 6(a) was generated by taking the mean of the results on five different corpora sampled form Wikipedia. The maximal standard deviation was 10.1 for OSDL, 9.3 for LDA, and 14.3 for LSA.

Out of the three topic models, LDA performs the best, with OSDL following closely behind. Latent semantic analysis does not seem applicable to word puzzle generation: it produces very few consistent sets.

The methods perform better on Wikipedia than on the NIPS proceedings. There may be two reasons for this: each corpus of Wikipedia articles contains 10,000 documents, while the corpus of NIPS proceedings contains only 1740. Furthermore, the similarity measure used (i.e., Explicit Semantic Analysis) relies on Wikipedia as background knowledge (see Section 6.3.2).



(a) Wikipedia (b) NIPS conference proceedings

Figure 6. Number of consistent sets produced by the different topic models, as the threshold δ grows. The two figures show the two corpora, Wikipedia articles and NIPS proceedings

6.3. Word puzzles

In this section, we take a look at the word puzzles generated from the two corpora: the one sampled from Wikipedia and the NIPS proceedings. As we found in the previous section that LSA is not capable of producing enough consistent sets, we only generate topics with LDA and OSDL. Furthermore, according to our experiments, the puzzles generated using OSDL and LDA are indistinguishable, so the two algorithms are not treated separately.

The presented method is capable of generating a great many puzzles, even from a single corpus. As the topic models used (i.e., LDA and OSDL) are randomly initialized, they can be used multiple times to generate different topics, therefore, different *consistent sets*. The unrelated elements mixed with these sets can also vary. We demonstrate the capabilities and pitfalls of the method by selected examples.

Based on the observations in the previous section, we chose $\delta = 0.1$ in every experiment to obtain a significant number of good enough consistent sets. The parameters of the topic models are the same as in the previous section.

In the next two sections, we examine a corpus consisting of $10,000$ documents sampled from Wikipedia. Beginner and intermediate level puzzles are produced. As *odd one out* and *choose the related word* puzzles are generated nearly identically, they are not discussed separately. We included some of the generated choose the related word puzzles on Table 4. In Section 6.3.3, puzzles that cover a narrow domain are generated from the NIPS proceedings.

6.3.1. Beginner puzzles

Beginner puzzles (Table 1) can be solved at first glance by a person who understands the language and has a wide vocabulary, for example, the puzzles *vote, election, candidate, voters, sony*, or *olympic, tournament, world, championship, acid*. These could be useful for e.g., beginner language learners or, with a suitable corpus, for children.

The parameters for generating these puzzles are $\theta = 0.02$, and $\phi = 0.005$ (see Section 5). In other words, the puzzles generated consist of a consistent set of related words and an unrelated word.

Some puzzles require specific knowledge about a topic. To solve the puzzles *harry, potter, wizard, ron, manchester* and *superman, clark, luthor, kryptonite, division*, the solver must be familiar with the book, film, comic, etc. To solve *austria, german, austrian, vienna, scotland*, geographic knowledge is needed.

Stemming can introduce errors. In the puzzle *animals, manga, released, japanese, tournament*, the words *animals* and *anime* were both incorrectly stemmed to the stem *anim*. The puzzle generation process is unable to dis-

tinguish them, hence the word *animals* is mistaken for the word *anime*. Even correct stemming can result in erroneous puzzles, as the puzzle *line, training, service, rail, orchestra*.

Consistent set of words				Odd one out
vote	election	candidate	voters	sony
line	training	service	rail	orchestra
church	orthodox	presbyterian	evangelical	buddhist
olympic	tournament	world	championship	acid
animals	manga	released	japanese	tournament
austria	german	austrian	vienna	scotland
devil	demon	hell	soul	boat
harry	potter	wizard	ron	manchester
superman	clark	luthor	kryptonite	division
magic	world	dark	creatures	microsoft

Table 1. Odd one out – beginner puzzles

Separate the topics puzzles are beginner puzzles by definition. In order for the two sets of words to be separable, they must differ considerably in meaning. So they are generated using the same parameters, $\theta = 0.02$, and $\phi = 0.005$. Table 2 contains some of the puzzles produced.

Consistent set 1	Consistent set 2
plant, tree, seed, garden	irish, ireland, dublin, patrick
water, heat, temperature, pressure	superman, clark, luthor, kryptonite
car, vehicles, engine, ford	church, saint, orthodox, christian
russian, russia, moscow, soviet	patients, treatment, cancer, disease
king, prince, queen, royal	chemical, acid, compounds, reaction
jump, fence, horse, rider	moon, orbit, planet, sun
cell, gene, protein, disease	harry, potter, voldemort, horcrux
band, album, tour, released	church, catholic, bishop, pope
navy, ship, naval, fleet	receptor, cell, peptide, stimulation
language, dialect, linguistic, spoken	club, league, cup, season

Table 2. Separate the topics puzzles

6.3.2. Intermediate puzzles

Among the beginner puzzles (Table 1) there is a puzzle where multiple solutions are possible, but the overall composition of the puzzle favors a single word. In the puzzle *church, orthodox, presbyterian, evangelical, buddhist*, both *church* and *buddhist* could be the *odd one out*, but, as four of the words are strongly related to christianity, the odd one out is *buddhist*. These *intermediate puzzles* are harder to solve, and feel more natural to solvers who know the language.

Consistent set of words				Odd one out
cao	wei	liu	emperor	king
superman	clark	luthor	kryptonite	batman
devil	demon	hell	soul	body
egypt	egyptian	alexandria	pharaoh	bishop
singh	guru	sikh	saini	delhi
language	dialect	linguistic	spoken	sound
mass	force	motion	velocity	orbit
voice	speech	hearing	sound	view
athens	athenian	pericles	corinth	ancient
data	file	format	compression	image
function	problems	polynomial	equation	physical

Table 3. Odd one out – intermediate puzzles

Intermediate puzzles are generated by introducing an additional constraint: the *odd one out* word should be related to the words in the consistent set. This is achieved by increasing ϕ: we set the parameters to $\phi = 0.1$ and $\theta = 0.2$. At first, it would seem counterintuitive that parameter θ is larger than δ. However, as the method is already constrained by the topic model and the set the unrelated words can be chosen from, we found that often, the *odd one out* word will be weakly related to the others that form a cohesive whole.

Although the presented method is based on semantic similarity, it is able to create surprisingly subtle puzzles. In the puzzle *voice, speech, hearing, sound, view*, the word *view* has a different modality than the others. To solve the puzzle *cao, wei, liu, emperor, king*, the solver should be familiar with the three kingdoms period of chinese history. For *egypt, egyptian, alexandria, pharaoh, bishop*, knowledge of Egyptian history, for *athens, athenian, pericles, corinth, ancient*, familiarity with the Peloponnesian War is required. In *singh, guru, sikh, saini, delhi*, all the words except *delhi* are related to sikhism. The puzzle *function, problems, polynomial, equation, physical* can be solved only with a basic knowledge of mathematics and physics.

Examples			Candidate words		
museum	collection	library	**history**	march	troops
division	tank	corps	**armoured**	united	field
book	published	public	**author**	science	organization
energy	particles	quantum	**physical**	process	future
regiment	battalion	army	**infantry**	service	king
devil	demon	hell	**soul**	love	man
story	short	fiction	**tales**	newspaper	script
football	club	coach	**cup**	united	university
bulgarian	bulgaria	turkish	**byzantine**	army	ancient
court	constitution	amendment	**rights**	organization	voters

Table 4. Choose the related word – intermediate puzzles

6.3.3. Word puzzles from narrow domains

In this section, we examine the word puzzles generated from a corpus that contains documents from a relatively narrow domains: the corpus of NIPS proceedings.

This corpus is harder to utilize because the similarity measure used (i.e., Explicit Semantic Analysis) relies on Wikipedia as background knowledge. Because of that, words that are not present or are very rare in Wikipedia are automatically excluded from the puzzles. As Explicit Semantic Analysis can work with different corpora, Wikipedia could be exchanged, provided that one has access to a large enough corpus. However, such corpora are generally hard to acquire. As demonstrated on Table 5, good results can be obtained by using Wikipedia as background knowledge.

Consistent set of words				Odd one out
prior	bayesian	probability	posterior	effect
continuous	discrete	function	space	processing
network	sigmoid	neural	feedforward	system
code	encoding	decoding	bit	developed
gaussian	distribution	covariance	variance	data
rule	based	system	knowledge	phase
model	data	parameters	distribution	theory
current	voltage	circuit	transistor	signal
auditory	sound	cochlear	cochlea	small
neurons	network	activity	connections	learning

Table 5. Odd one out – domain-specific intermediate puzzles

7. Conclusion

We have proposed a knowledge-lean method to generate word puzzles from unstructured and unannotated corpora. The presented method is capable of generating three types of puzzles: odd one out, choose the related word, and separate the topics. The difficulty of the puzzles can be adjusted.

A topic model is used to generate a collection of topics. Consistent sets of related words are produced from these topics by an algorithm based on network capacity and semantic similarity. The puzzles are produced by mixing these sets with weakly related elements: words or other consistent sets.

The results showed that the system produces high-quality puzzles, whose solution is clear and unique. Puzzles of two difficulty levels were generated: beginner and intermediate. Beginner puzzles could be suitable for, e.g., beginner language learners. Intermediate puzzles require more, often specific knowledge to solve. Domain-specific puzzles were generated from a corpus of NIPS proceedings.

The presented method is capable of helping puzzle designers compile a collection of word puzzles in a semi-automated manner. In this setting, the method is utilized to produce a great number of puzzles. Puzzle designers can choose and maybe modify the ones they want to include in the collection.

References

[1] *PCGames '11: Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, New York, NY, USA, 2011. ACM.

[2] **Alsumait, L., D. Barbar, J. Gentle and C. Domeniconi**, Topic Significance Ranking of LDA Generative Models, *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, ECML-PKDD 09, Berlin, Heidelberg, 2009. Springer-Verlag, 67–82.

[3] **Ashlock, D.**, Automatic generation of game elements via evolution, *Computational Intelligence and Games (CIG)*, Dublin, Aug 2010, 289–296.

[4] **Blei, D. M., Andrew Y. Ng and M. I. Jordan**. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, **3** (2003), 993–1022.

[5] **Bloehdorn, S., R. Basili, M. Cammisa and A. Moschitti**, Semantic Kernels for Text Classification Based on Topological Measures of Feature Similarity, *Data Mining, 2006. ICDM '06. Sixth International Conference on*, 808–812.

[6] **Carter, P.**, *IQ and Psychometric Test Workbook*, Kogan Page Business Books, London, UK, 2005.

[7] **Choi, F., P. Wiemer-Hastings and J. Moore**, Latent Semantic Analysis for Text Segmentation, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, 2001, 109-117.

[8] **Colton, S.**, Automated Puzzle Generation *In Proceedings of the AISB'02 Symposium on AI and Creativity in the Arts and Science*, 2002.

[9] **Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman**, Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, **41** (1990), 391–407.

[10] **Doran, J. and I. Parberry**, A prototype quest generator based on a structural analysis of quests from four (MMORPGs), *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, New York, NY, USA, 2011. ACM., 1–8.

[11] **Egozi, O., S. Markovitch and E. Gabrilovich**, Concept-Based Information Retrieval Using Explicit Semantic Analysis, *ACM Transactions on Information Systems*, **29** (2) (2011), 1–34.

[12] **Eppstein, D.**, Nonrepetitive Paths and Cycles in Graphs with Application to Sudoku, *CoRR*, 2005.

[13] **Gabrilovich, E. and S. Markovitch**, Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis, *International Joint Conferences on Artificial Intelligence* 2007, 1606–1611.

[14] **Gabrilovich, E. and S. Markovitch** Wikipedia-based Semantic Interpretation for Natural Language Processing, *Journal of Artificial Intelligence Research (JAIR)*, **34** (2009), 443–498.

[15] **Gurevych, I., C. Müller, T. Zesch**, What to be? – Electronic Career Guidance Based on Semantic Relatedness, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2007, 1032–1039.

[16] **Hofmann, T.**, Probabilistic latent semantic indexing, *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, New York, NY, USA, 1999, 50–57.

[17] **Hu, T. C.**, Letter to the Editor – The Maximum Capacity Route Problem *Operations Research*, **9** (6) (1961), 898–900.

[18] **Iosup, A.**, POGGI: generating puzzle instances for online games on grid infrastructures, *Concurrency and Computation: Practice and Experience*, **23** (2) (2007), 158–171.

[19] **Jungnickel, D.**, *Graphs, Networks and Algoritms*, Springer, 3rd edition, 2007.

[20] **Kruskal, J. B.**, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings of the American Mathematical Society*, **7** (1) (1956), 48–50.

[21] **Mohler, M. and R. Mihalcea** Text-to-text semantic similarity for automatic short answer grading, *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 2009, 567–575.

[22] **Porter, M. F.**, An algorithm for suffix stripping, *Readings in information retrieval*, 1997, 313–316.

[23] **Radinsky, K., E. Agichtein, E. Gabrilovich and S. Markovitch**, A Word at a Time: Computing Word Relatedness using Temporal Semantic Analysis, *Proceedings of the 20th International World Wide Web Conference*, (2011), 337–346.

[24] **Rosen-Zvi, M., T. Griffiths, M. Steyvers and S. Padhraic** The author-topic model for authors and documents, *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, AUAI Press, 2004, 487–494.

[25] **Simonis, H.**, Sudoku as a constraint problem, *Proc. 4th Int. Works. Modelling and Reformulating Constraint Satisfaction Problems*, 2005, 13–27.

[26] **Sorg, P. and P. Cimiano**, Cross-lingual Information Retrieval with Explicit Semantic Analysis, *Working Notes for the CLEF 2008 Workshop*, 2008.

[27] **Steyvers, M. and T. Griffiths**, Probabilistic Topic Models, *Handbook of Latent Semantic Analysis*, Lawrence Erlbaum Associates, 2007.

[28] **Szabó, Z., B. Póczos and A. Lőrincz**, Online Group-Structured Dictionary Learning, *IEEE Computer Vision and Pattern Recognition (CVPR-2011)*, 2865–2872.

[29] **Verguts, T. and P. D. Boeck**, A Rasch Model for Detecting Learning While Solving an Intelligence Test, *Applied Psychological Measurement*, **24** (2) (2000), 151–162.

[30] **Zobel, Justin and A. Moffat**, Exploring the similarity space, *SIGIR Forum*, **32** (1998) 18–34.

B. Pintér, Gy. Vörös, Z. Szabó and A. Lőrincz
Department of Software Technology and Methodology
Eötvös Loránd University
H-1117 Budapest, Pázmány P. sétány 1/C
Hungary
bli@elte.hu, vorosgy@inf.elte.hu
szzoli@cs.elte.hu, andras.lorincz@elte.hu