Text Categorization with Semantic Commonsense Knowledge

X X, Y Y

XXXXXXXXXXXXXXXXX,
XXXXXXXXXXXXXXXXX,
{x, y}@zzz

Abstract. Most of text categorization research exploit bag-of-words text representation. In this approach, however, all contextual information contained in text is neglected. Therefore, capturing semantic similarity between text documents that share very little or even no vocabulary is not possible. In this paper we present an approach that combines well established kernel text classifiers with external contextual commonsense knowledge. We propose a method for computing semantic similarity between words as a result of diffusion process in ConceptNet semantic space. Evaluation on a Reuters dataset show a substantial improvement in precision of classification.

1 Introduction

Text is the primary medium of representing and distributing information. Categorization is one of the basic methods of organizing textual data. Research on this topic have been dominated with machine learning approach and a predominant number of papers focus on kernel methods [1]. In most reported works text were represented with a Vector Space Model (a.k.a. bag-of-words) and similarity between two pieces of text were computed as a function of words shared by the two. This assumption, however, makes it very hard to capture any semantic similarity between text documents that share very little or even no vocabulary. This issue was addressed with statistical and algebraic tools [2]. On the other hand, there is a long history of works evaluating word relatedness using hierarchical semantic network representations that gets back to 1960s [3]. More recent works concentrate on WordNet [4] as a primal source of information on relations between words (e.g. [5][6]). In this paper we present an approach that tries to combine well established kernel text classifiers with contextual common-sense knowledge brought by ConceptNet [7]. We propose a method for computing semantic similarity between words that is backed by the common-sense relations graph. The similarity is expressed as a result of diffusion process. Resulting kernel function is later evaluated in a text categorization problem.

This paper is organized as follows. In next section we briefly describe linear kernels for text classification problems. In the third section we define proximity of words and outline methods for construction word proximity matrices. In

fifth section we describe in details our diffusion algorithm. Results of experimental evaluation are presented in sixth section. Finally, we conclude and give an outlook for future works in the last section.

2 Linear Kernels for Text

Simple linear kernels perform very well in text categorization problems [1]. They implement an IR-developed Vector Space Model (VSM) using a mapping $\phi(d)$ that embeds a piece of text d into a space where each dimension corresponds to one of N words in dictionary $\mathcal{D}$. Order in which words appear in text is neglected. Similarity between two pieces of text, $d_1, d_2 \in Docs$, is computed as a dot product of their embeddings,

$$\kappa(d_1, d_2) = \langle \phi(d_1), \phi(d_2) \rangle = \boldsymbol{d}_1 \boldsymbol{d}_2^T.$$

The more words documents have in common, the more similar they are.

Due to simplistic assumption, however, linear kernels are not able to capture any semantic similarity between text documents that share very little or even no vocabulary. This problem can be addressed by extending from VSM to GVSM [8] [9]. *Semantic linear kernels* [10] implement this extension, and incorporate additional information on words similarity. They are defined as,

$$\kappa_s(d_1, d_2) = \langle \phi_s(d_1), \phi_s(d_2) \rangle = \phi(d_1) \boldsymbol{S} \boldsymbol{S}^T \phi(d_2)^T. \tag{1}$$

where $\boldsymbol{S}$ is a *semantic matrix*. The $\boldsymbol{S}$ matrix could be any $N \times k$ matrix with all positive entries that captures semantic similarity between words. If k is equal to N, the semantic matrix can be regarded as a word-to-word similarity matrix. Indeed, the s_{ij} entry would express semantic similarity between i-th and j-th words in a dictionary. When k is smaller then N, this leads to some dimensionality reduction step, so that subsequent computations are performed in this k-dimensional, reduced space.

3 Proximity of Words

Proximity is usually defined with help of a distance function, $\Delta : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ (e.g. [6]). A proximity is inverse of a distance. Two objects are given a large proximity value if they are in a close distance. If they are the same, proximity is equal to 1. With help of a distance function, proximity is given as

$$p(u, v) = \begin{cases} inv(\Delta(u, v) + 1) & \Longleftrightarrow \Delta(u, v) \neq \infty, \\ 0 & \Longleftrightarrow \Delta(u, v) = \infty, \end{cases} \tag{2}$$

where $inv : [1, \infty) \to [0, 1]$ is some monotonically decreasing function, e.g. $1/x$ or $\exp(-x)$. A proximity function can also be constructed without explicit computation of a distance between words. It can be generalized to any monotonically decreasing function $p : \mathcal{D} \times \mathcal{D} \to [0, 1]$ that captures our intuitive proximity

of their meaning. By constructing such a function we define a corresponding distance function implicitly.

To construct proximity matrix as in (1) we need

- source of information on relations between words,
- method for computing proximity and relevance of words.

In most approaches the training data itself is used to extract relations between words, with common co-occurrence being the most frequent indicator of their relatedness. On the other hand there are relatively few works that involve external sources of information of (e.g. [5], [6]). Methods to obtain proximity values are closesly related to a source of data used. For structured representations, usually graphs, diffusion methods can be applied [2] [11].

In this paper we investigate application of external data and a novel method of proximity computation. In two subsequent sections we discuss in details the source of relations on words proximity and proposed diffusion algorithm.

4 ConceptNet as a Source of Information on Relations between Words

Firstly, we should decide what properties a good source of information on words relations should posses? Since we are analyzing general text classification problem it should cover wide scope of topics. Moreover, it should contain relations and associations between words, that are typical and obvious to humans but not clear to machines. At the same time, it should include sophisticated vocabulary that turns out to make the most discriminative features. In fact, there are only very few publicly available sources of structured information that could be used for this purpose.

The best known and used in predominant number of works is WordNet [4]. This hand-crafted lexical database system consist of well structuralised information organized within part of speech groups. WordNet's recent version lacks, however, relations between these groups. For instance, it does not provide obvious information that a "dog" "barks", but comes with a detailed mammals taxonomy. For context aware processing such information are of a little value, though. Additionally WordNet requires an disambiguation step to map analyzed words to senses.

We argue that for classification problem information on contextual rather then structural relations is of greater need. ConceptNet [7], made publicly available recently, is a semantic network designed for commonsense contextual reasoning. It was automatically built from a collection of 700,000 sentences, a corpus being a result of collaboration of some 14,000 people. It provides commonsense contextual associations not offered by any other knowledge base. ConceptNet traded the precise relations given by WordNet for noisy and imprecise yet obvious and valuable for humans. For instance, ConceptNet lacks any fine information on mammals taxonomy that WordNet provides, but comes with an obvious hint that a "dog" is a "pet".

ConceptNet is organized as a massive directed and labelled graph. It is made of about 300,000 vertexes and about 1.5 million edges, corresponding to words or phrases, and relations between them, respectively. Vast part of the vertexes represent common actions or chores given as phrases, e.g. "drive a car" or "buy food". There are approx. 260,000 vertexes of this type. The remaining nodes are single words (including stopwords). There are also six types of relations (or edges) between vertexes. The largest connected component encompasses almost the whole ConceptNet graph. Its structure is also a bit bushy, with a group of highly connected nodes, and "person" being the most connected, having indegree of about 30,000 and outdegree of over 50,000. There are over 86,000 leaf nodes and approximately 25,000 root nodes (nodes that have no incoming edges). An average degree of a node is 4.69.

5 Proximity Evaluation

Having the structure of ConceptNet in mind, we propose an alternative proximity function defined on graph vertexes. General idea of the algorithm is also inline with classic psycholinguistic theory of spreading activation [12] in semantic memory. We have chosen the following objectives for its construction:

- proximity decreases with number of visited nodes,
- vertexes connected directly or through some niche links are in a short distance, hence they are proximate,
- connections going through highly connected nodes increase ambiguity, therefore proximity should be inversely proportional to number of nodes that could be visited within given number of steps,
- computational complexity should be low.

An algorithm constructed according to these rules is presented in the following.

Outline of the Algorithm Here we propose an algorithm that computes proximity of words basing on a graph structure. It should be noted that this procedure is not symmetric, i.e. $p(t_1, t_2)$ does not necessarily have to be equal to $p(t_2, t_1)$, depending on a graph.

Our procedure is made on an assumption that the proximity is proportional to amount of some virtual substance that reaches the destination node v as a result of injection to node u, followed by diffusion through graph edges. The diffusion process is governed according to the following simple rules,

1. at every node the stream splits into smaller flows proportional to number of edges going out from the node,
2. at every junction, a fraction ϱ of the substance sinks in and does not go any further,
3. if some edge points to a node that has been visited before, we assume that the node is saturated and can not take in any more substance.

The process continues until all reachable nodes "get wet".

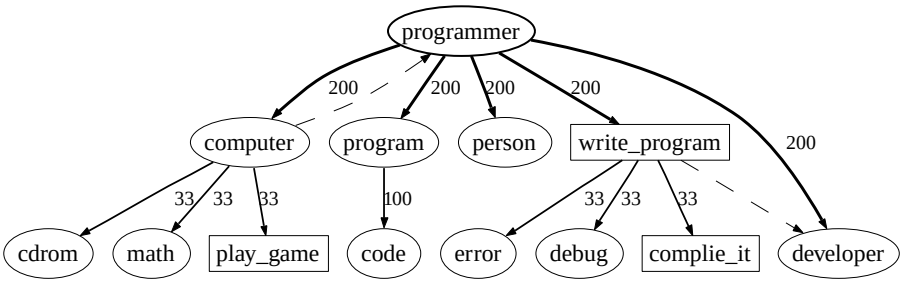

Fig. 1. A fragment of ConceptNet graph; rounded nodes correspond to single words and rectangular nodes represent phrases, all types of relations were collapsed into a single one.

Description After this somewhat informal introduction, we will discuss the algorithm in details and give relationale for particular design choices that justify our approach. We will use ConceptNet's fragment given in Fig. 1 as an illustration. In this illustrative example we compute proximity of the `programmer` node to the rest of the nodes. In the kick off step, the `programmer` has been injected with 2,000 units of some fluid.

The first rule of the diffusion process assures that proximity of a node to its direct neighbors is proportional to number of neighbors. Consequently, highly connected nodes are given some penalty. On the other hand, if there are very few outlinks, they are considered much more informative. The `programmer` (see Fig. 1) has five adjacent nodes, therefore each of them is flooded with one fifth of the amount of the substance available to `programmer`. The purpose of this rule is to diminish proximity of nodes reachable through some highly connected nodes. For instance, ConceptNet's "person" is connected to about 50,000 other nodes. Clearly, any association going through this node can not be considered informative or unique. This rule assures that any connection going through the "person" node will yield very low proximity to the source.

The second rule of the diffusion process was introduced to decrease proximity with number of intermediate nodes required to reach the destination node. Since with every junction risk of drifting off the topic increases, this rule requires paths connecting proximate nodes to be as short as possible. In our example, at every intermediate node half of the substance is retained (ϱ is equal to 0.5), and only the other half diffuses further through edges. If this rule was omitted `programmer` would be equally related to `program` and `code`. While this turns out to be quite accurate, such relation does not hold in general.

The purpose of the third rule is to assure monotonicy of the function with increasing length of a path connecting some nodes. By saturating a node we make sure that its proximity always takes fixed value that is smaller then the amount of substance injected to the source node. This rule also assures that the algorithm terminates after a finite number of steps. If this rule were to be

suspended, every node reachable with more then one walk would be boosted with every such connection. Popular nodes that are members of many walks would be pumped-up to infinity. On the other hand, the saturation rule seems to be in line with the way humans associate words — our first guesses are usually the most accurate. In Fig. 1 edges pointing to saturated nodes are dashed. They are not taken into account when a degree of a node is computed. For instance, `computer` has four outgoing edges. However, since an edge pointing back to `programmer` is neglected, the effective degree of `computer` is decreased to three.

There is one more point to be discussed about the algorithm. We have not decided yet on the order in which nodes and edges are visited during diffusion process. We propose to process a graph in order of node's proximity values. This process will mimic a wave surging in a pipe system — the strongest wave goes furthest. To implement this flow we pick nodes in order of their proximity — proximate vertexes go first. If some nodes are equally proximate, then the degree decides and nodes having lower number of outgoing edges are given a privilege. However, if there is more then one node of the same degree, a conflict arises. We show how to resolve it in the next paragraph.

Conflicts Resolution As we have said the vertexes are picked in order of their proximity to the source node. A problem arises when there is more then one equally proximate vertex. In fact this is the situation that always occurs immediately after the initial step of the algorithm, because the kick-off injection is always split equally between nodes adjacent to the source. We have chosen degree of a node as the second criterion — the lower the better. However, what to do when this is not conclusive and there is still more then one possibility?

To solve this problem we propose to process all conflicting nodes in a single step. As a 'single step' we understand that nodes adjacent to the conflicting vertexes are not saturated until the last of them is processed. The order in which nodes are processed might change proximity results only if it affects effective degrees of nodes. It will not happen if the saturation is postponed. There are two cases to be analyzed — when conflicting vertexes share no adjacent nodes and the other situation when there are some shared adjacent nodes.

Clearly, if there are no shared adjacent nodes the effective degree of any of conflicting nodes will not be affected by any other. Therefore, all nodes adjacent to S yield equal proximity of $\frac{p'\varrho}{d}$.

In the other case, there are some nodes that share adjacent nodes. Now by applying the normal processing routine that involves immediate saturation to some node, we would decrease the effective degree of some of the remaining nodes. Although, if saturation of edges is postponed until the last of the conflicting nodes is processed, effective degree of any node will not be changed. Hence, the order in which the nodes are picked does not matter and all adjacent nodes are given equal proximity value of $\frac{p'\varrho}{d}$.

A way in which node are processed resemble Dijkstra SP-algorithm. The main difference are changed criteria used for picking a next node. Moreover, the

purpose is quite opposite — the algorithm is used not to compute SP-distances but to assign values to nodes.

Running Time Running time of a single diffusion operation using a standard binary heap as a backing data structure is $O((|V| + |E|)log|V|)$, where $|V|$ and $|E|$ are numbers of vertexes and edges of ConceptNet graph, respectively. Since, to construct complete proximity matrix we require results of diffusion operations for all words in the graph, total running time rises to $O(W(|V| + |E|)log|V|)$, where W is number of words. It should be noted that $W << |V|$ because word nodes are a small fraction of of the whole ConceptNet graph and most of nodes correspond to phrases.

6 Experiments

Here we want to investigate how does the algorithm presented above perform in practice and how does it compare to established text classifiers. In our experimental works we intended to answer the following questions:

(Q1) Do ConceptNet and the diffusion algorithm described above improve classification performance?
(Q2) Are there any cases where they can decrease performance? Why?
(Q3) What is the influence of diffusion parameter ϱ?

In order to answer them we conducted a series of experiments using a standard benchmark for text classification problems: the Reuters-21578 "ModApte split". We experimented only on ten most frequent categories, i.e. `acquisition`, `corn`, `crude`, `earn`, `grain`, `interest`, `money-fx`, `ship`, `trade` and `wheat`.

6.1 Preprocessing

ConceptNet as it comes is a bit noisy. For propose of our experiments we cleaned it a little. At first, all redundant entries were removed. Subsequently all entries containing digits and self-loops were deleted. Afterwords we extracted the largest connected component of the graph and removed other nodes. Remaining nodes were then indexed in a dictionary and divided into three groups — words, phrases and junk words. By a phrase we considered a string with white spaces. There are 260,954 nodes of this type. Words were stemmed with a Lovins stemmer [13] and ones sharing a common stem were collapsed down into a single node. As junk words we considered stopwords found on Weka [14] stoplist and words with stems shorter then four letters. There were 1,638 such words. The junk words were retained to keep connectivity of the graph. After this filtering procedure there were 20,227 words left that we consider features.

Documents of the Reuters corpus were indexed using two separated dictionaries. The first dictionary was built in a usual way from all words that occurred in a training set. The other dictionary was constructed from ConceptNet's feature words. We also removed terms that occurred in a training less then three

times. Documents indexed with these dictionares made two experimental data sets, $\mathbf{D}$ and $\mathbf{D}_{CN}$, respectively. Vectors were then normalized with L_1-norm.

6.2 Comparative Evaluation

In order to answer questions Q1 and Q2 we experimentally compared SVM classifiers built on diffusion proximity kernels and standard linear kernels. Experiments were conducted on $\mathbf{D}_{CN}$ and $\mathbf{D}$ datasets, while the latter was used as a baseline. In all experiments SVMs' C parameter was set to 1 and diffusion parameter ϱ to 0.5. Results obtained with a 10-fold cross-validation are given in Table 1.

Class	Precision			Recall			F_1		
	L-$\mathbf{D}$	L-$\mathbf{D}_{CN}$	DD-$\mathbf{D}_{CN}$	L-$\mathbf{D}$	L-$\mathbf{D}_{CN}$	DD-$\mathbf{D}_{CN}$	L-$\mathbf{D}$	L-$\mathbf{D}_{CN}$	DD-$\mathbf{D}_{CN}$
acq	0.954	0.903	0.943	0.946	0.905	0.740	0.950	0.904	0.829
corn	0.097	0.089	-	0.099	0.110	-	0.098	0.098	-
crude	0.848	0.841	-	0.776	0.630	-	0.810	0.720	-
earn	0.981	0.959	0.984	0.974	0.935	0.819	0.978	0.947	0.894
grain	0.303	0.315	0.366	0.127	0.150	0.025	0.179	0.204	0.047
interest	0.678	0.649	0.944	0.553	0.400	0.049	0.609	0.495	0.094
money-fx	0.707	0.672	0.719	0.745	0.687	0.143	0.725	0.679	0.239
ship	0.650	0.689	-	0.484	0.461	-	0.555	0.553	-
trade	0.854	0.810	1.000	0.826	0.724	0.057	0.840	0.764	0.107
wheat	0.086	0.033	-	0.103	0.004	-	0.094	0.008	-

Table 1. Results of experimental evaluation. Columns: L-$\mathbf{D}$ — linear kernel and $\mathbf{D}$ dataset; L-$\mathbf{D}_{CN}$ — linear kernel and $\mathbf{D}_{CN}$ dataset; DD-$\mathbf{D}_{CN}$ — diffusion kernel and $\mathbf{D}_{CN}$ dataset;

The proposed kernels improved precision while decreasing recall of the classifiers. For 5 out of 10 classes precision increased even over results obtained with a full vectors (L-$\mathbf{D}$ column). However, for 4 classes (marked with a "-") our method could not recognize any positive sample, and simply gradated into a majority voter. Moreover, the increase in precision came at a price of lower recall for classes that were learned correctly. Relatively high recall was reported only for `acq` and `earn` classes. We suspect that introduction of relations between words brought with ConceptNet allowed for disambiguation of words, and as a result improved precision of categorization. However, they also reduced the influence of individual words that were allowed to play a discriminative role in linear kernels. Low recall may also be due to a a weak fitting between the ConceptNet graph and Reuters corpus. The latter uses rather formal and precise vocabulary, while the former is organized around informal activities. We expect our algorithm to work much better in more casual text categorization tasks.

6.3 Diffusion Parameter ϱ

For $\varrho = 1$ the diffusion kernel coincides with a standard linear kernel and the diffusion process is not performed at all. At the other extreme, when $\varrho = 0$, the distribution of the substance depends only on the structure of the graph. To examine influence of the parameter ϱ on classification performance we conducted a series of experiments in a whole domain of the parameter, $\varrho \in [0, 1]$, with a 0.05 step. Results for selected classes obtained with 10-fold cross-validation for C set to 1 are presented in Fig. 2. The plots are rather flat with very small differences

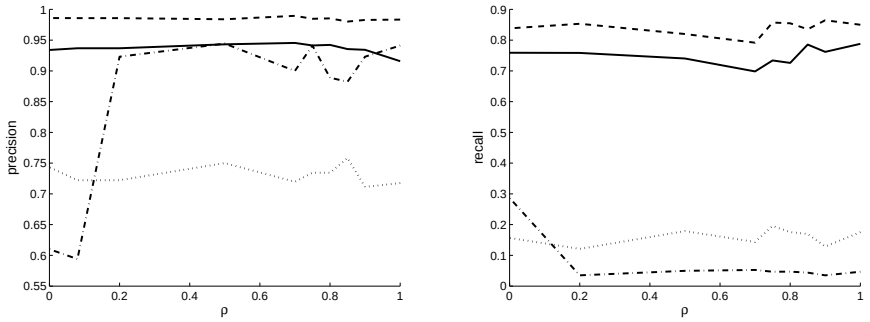

Fig. 2. Precision (left) and recall (right) of classifier in function of ϱ; Classes: `acq` (solid line), `earn` (dashed line), `interest` (dash-dotted line), `money` (dotted line)

in a whole range of the parameter with larger fluctuations at the extremes. This indicates that the influence of the diffusion parameter ϱ is limited and the proximity of words comes mostly from the structure of ConceptNet graph. Therefore, for practical usage ϱ could be set any value within its domain and does not introduce much additional burden to a learning task.

7 Conclusions

In this paper we presented an approach that combined kernel text classifiers with contextual common-sense knowledge brought by ConceptNet. Experimantal evaluation have shown that contextual relations contained in ConceptNet allowed for increased precision. This came, however, at a price of low recall. This might be due to limitations of the semantic space. In future, we plan to address this problem by extanding semantic space by combining ConcepNet, WordNet and Microsoft Mindnet [15] into a single graph.

References

1. Joachims, T.: Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Kluwer Academic Publishers (2002)

2. Kandola, J., Shawe-Taylor, J., Cristianini, N.: Learning semantic similarity. In: NIPS 15, Cambridge, MA, MIT Press (2003) 657–664
3. Collins, A., Quillian, M.: Retrieval from semantic memory. Journal of Verbal Learning and Verbal Behavior **8** (1969) 240–247
4. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: WordNet: An on-line lexical database. International Journal of Lexicography **3**(4) (1990) 235–312
5. Basili, R., Cammisa, M., Moschitti, A., Rome, I.: A Semantic Kernel to Classify Texts with Very Few Training Examples. Informatica **30** (2006) 163–172
6. Siolas, G., dAlché Buc, F.: Support Vector Machines Based on a Semantic Kernel for Text Categorization. In: Proceeding of IJCNN, IEEE Computer Society Washington, DC, USA (2000) 205–209
7. Liu, H., Singh, P.: Conceptnet – a practical commonsense reasoning tool-kit. BT Technology Journal **22**(4) (2004) 211–226
8. Jiang, F., Littman, M.L.: Approximate dimension equalization in vector-based information retrieval. In: ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2000) 423–430
9. Wong, S.K.M., Ziarko, W., Wong, P.C.N.: Generalized vector spaces model in information retrieval. In: SIGIR '85: Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (1985) 18–25
10. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
11. Kondor, R.I., Lafferty, J.: Diffusion kernels on graphs and other discrete structures. In: Proceedings of MGTS at ECML/PKDD. (2002) 315–322
12. Collins, A., Loftus, E.: A spreading-activation theory of semantic processing. Psychological Review **82**(6) (1975) 407–428
13. Lovins, J.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics **11** (1968) 22–31
14. Witten, I., Frank, E., Trigg, L., Hall, M., Holmes, G., Cunningham, S.: Weka: Practical Machine Learning Tools and Techniques with Java Implementations. (Department of Computer Science. University of Waikato. New Zealand)
15. Vanderwende, L., Kacmarcik, G., Suzuki, H., Menezes, A.: Mindnet: an automatically-created lexical resource. In: Proceedings of HLT/EMNLP on Interactive Demonstrations, Morristown, NJ, USA, Association for Computational Linguistics (2005) 8–9