Reasoning by analogy for fact extraction

Serge V. Slipchenko

Department of Neural Information Processing Technologies
International Research and Training Center for
Information Technologies and Systems
Serge.Slipchenko@gmail.com

Abstract

Method of fact extraction based on reasoning by analogy with binary distributed representations is proposed. Details of building binary distributed representations and application of reasoning by analogy for typed-dependency trees outlined.

1. Introduction

Automatic knowledge extraction from natural language texts is an important direction of computer linguistics and artificial intelligence research. In conditions of fast development of global Internet network and growing amount of available documents manual search and processing of information require huge human and financial resources. Yet only a small fraction of information is processed. Therefore interest to automatic knowledge extraction from natural language texts keeps growing. Examples of knowledge of interest are descriptions of geo-politic events [1], results in bio-molecular research [2], or object relationships in news [3].

Models and methods that are used for relation extraction vary from simple classification models (e.g., combination of CRF and Bayesian rules [1], or SVM with kernels on dependency trees [2]) to complex probabilistic graphical models (e.g., Markov Logic [3] that combines Markov networks and first-order logic calculus).

The proposed approach is a development of ideas and methods of case-based reasoning [4-7], which are characterized by:

- Use of a sample database to build inferences that allows simple interpretation of the inference process.
- Use of binary distributed representations of a probe and the samples that in contrast to logical and non-distributed models allow fast and effective parallel lookup of samples from the database.

Methods that are based on binary distributed representations were used successfully in modeling reasoning by analogy [4-6], but text descriptions of samples were formalized by experts and rewritten to structured machine-readable descriptions [7,8]. Automatic construction of structured machine-readable descriptions is the open research topic. This work focuses on simple fact extraction (much like the related works [1-3]), but is using methods of reasoning by analogy [4-6].

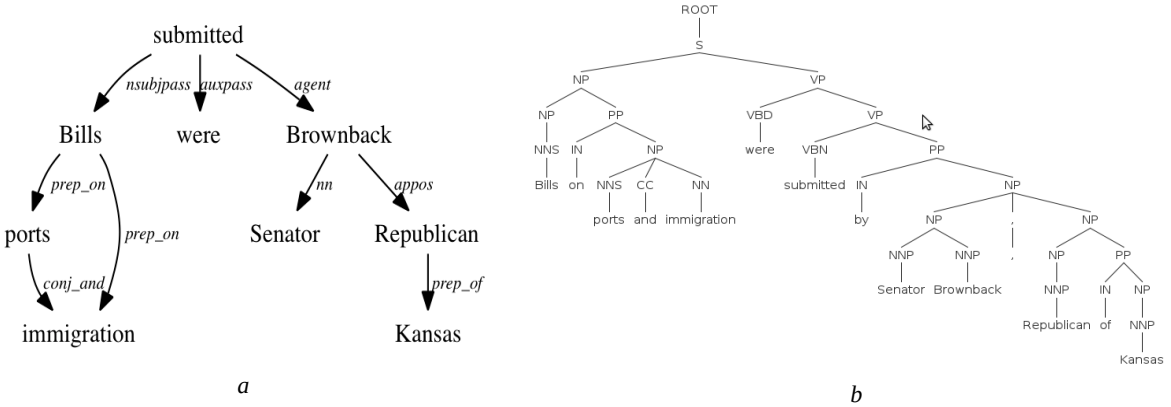
2. Fact extraction

This work is using Stanford typed-dependencies "trees", which really are Direct Acyclic Graphs (DAG) that are obtained by preprocessing constituency-based parsing tree [9]. Similar preprocessing is used to build syntactic relations in classification [2] and probabilistic graphical models [3].

2.1. Typed-dependencies and matching relations

Typed-dependency trees represent syntactic and semantic relations of words in a sentence (terminals only), while constituency-based parsing tree creates new non-terminal entities to represent the syntactic structure (see Figure 1 for details). Typed-dependencies are represented by triples *name (governing word, dependent word)*. E.g., for the sentence "**Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas**" we have the following triples: *nsubjpass*(*submitted*,*Bills*), *auxpass* (*submitted*,*were*), *agent*(*submitted*,*Brownback*), *nn*(*Brownback*,*Senator*), *appos*(*Brownback*,*Republican*), *prep_of*(*Republican*,*Kansas*), *prep_on*(*Bills*,*ports*), *conj_and*(*ports*,*immigration*) and *prep_on*(*Bills*,*immigration*).

The key benefit of typed-dependency trees for domain experts is a natural representation of relations by graph fragments, e.g. for the relation *submit*('*Brownback*', '*Bills on ports and immigration*') name is the general form of verb *submitted*, and object '*Brownback*' and subject '*Bills on ports and immigration*' are direct dependents of the verb. Therefore the problem of relation extraction can be stated as matching of probe graph fragments to sample graph fragments [2].



a *b*

Figure 1: Typed-dependency (a) and constituent-based (b) parse trees

Unfortunately, the complexity of the matching for the dependency tree kernel [2] is O (mn^3), where m – number of nodes in the probe, n – number of nodes in the sample.

2.2. Binary distributed representations

Graph embeddings in vector spaces [10-11] allows to cope the complexity problem by replacing an expensive graph similarity function with cheap vector similarity functions.

Binary distributed representations employ a similar vector-based approach to graph matching problem. Any graph element (node, edge or subgraph) is represented by a binary vector $\mathbf{X} \in \{0,1\}^N$ with a large number of elements N, but only a small fraction of non-zero elements $M/N << 1/2$. Similar items should have similar binary representations, whereas items with undefined similarity have dissimilar representations [4]. The measure of similarity is dot product of binary representations

$$(\mathbf{X},\mathbf{Y}) = \sum_{i=1,N} X_i Y_i \tag{1}$$

In the context of the graph matching we expect that similar subgraphs should have similar binary representations. This is an ill posed problem because there is no "good" theoretical measure of graph similarity like subgraph isomorphism or maximum common subtree that corresponds to the expert notion of graph similarity in the text mining domain.

Proper definition of the similarity measure should employ some learning process, but this definition is beyond the scope of this work.

2.3. Context-Dependent Thining

Binary distributed representations use unique random binary vectors to represent dissimilar atomic items. Complex items are built from atomic items using context-dependent thinning procedure (CDT) that applies to the set of binary vectors {**A**,**B**,…} and creates a new binary vector ⟨**A**,**B**,..⟩. CDT has properties that are important for building binary representations of graphs (see [12] details):

- Binary vectors of similar sets are similar: ⟨**A**,**B**,**C**⟩ ≈ ⟨**A**,**B**,**C**⟩.
- Binary vectors of dissimilar subsets are dissimilar: ⟨⟨**A**,**B**⟩,**C**⟩ ≠ ⟨**A**,⟨**B**,**C**⟩⟩.

Algorithm of the context-dependent thinning is the following:

1. The binary disjunction of the input vectors {$\mathbf{X}_i$} is built:

$$\mathbf{Z} = \vee_{i=1,n} \mathbf{X}_i \tag{2}$$

2. Binary conjunctions of the vector **Z** and its permutation $\tilde{\mathbf{Z}}_k$ combined K times:

$$\langle \mathbf{Z} \rangle = \vee_{k=1,K} (\mathbf{Z} \wedge \tilde{\mathbf{Z}}_k). \tag{3}$$

2.4. Binary distributed representations of typed-dependencies

Stanford typed-dependencies "trees" are Direct Acyclic Graphs (DAG). The structure of a typed-dependency graph is quite similar to the structured analog descriptions [4] that allows to use the modified "role-filler" scheme.

Structured analog description graph nodes represent relations, attributes and entities. The ordering of relation node children imposes implicit edge labels. Typed-dependency graph nodes represent words that may represent relations, attributes and entities, but there is no direct mapping. In order to overcome formalization difficulties the following procedure is proposed:

1. Create a unique binary random vector for every word in the vocabulary. E.g., $\mathbf{V}_{w(\text{submitted})}$, $\mathbf{V}_{w(\text{bills})}$, etc.
2. Create a unique binary random vector for every unique named entity in the sentence. E.g., $\mathbf{V}_{w(\text{Brownback})}$, $\mathbf{V}_{w(\text{Kansas})}$, etc.
3. Create a unique binary random vector for governor $\mathbf{V}^{dep}_{dependency}$ and dependant $\mathbf{V}^{gov}_{dependency}$ roles for every dependency relation *dependency*. E.g., $\mathbf{V}^{dep}_{agent}$, $\mathbf{V}^{gov}_{agent}$,etc.
4. Use CDT procedure to build a representation of the dependency relation:

$$\mathbf{V}^{dependency}_{w_{gov},n_{dep}} = \left\langle \begin{array}{l} \left\langle \mathbf{V}^{gov}_{dependency} \vee \mathbf{V}_{w_{gov}} \right\rangle \\ \vee \\ \left\langle \mathbf{V}^{dep}_{dependency} \vee \mathbf{V}_{n_{dep}} \right\rangle \end{array} \right\rangle \tag{4}$$

where $\mathbf{V}_{w_{gov}}$ - random vector of the governing word, $\mathbf{V}_{n_{dep}}$ - CDT vector of the dependent node. E.g,

$$\mathbf{V}^{agent}_{w(submitted),n(Brownback)} = \left\langle \begin{array}{l} \left\langle \mathbf{V}^{gov}_{agent} \vee \mathbf{V}_{w(submitted)} \right\rangle \\ \vee \\ \left\langle \mathbf{V}^{dep}_{agent} \vee \mathbf{V}_{n(Brownback)} \right\rangle \end{array} \right\rangle$$

5. Use disjunction of dependency relation representations to build a representation of the governing node. E.g.,

$$\mathbf{V}_{n(submitted)} = \mathbf{V}^{nsubjpass}_{w(submitted),n(Bills)} \vee \mathbf{V}^{nsubjpass}_{w(submitted),n(Brownback)}$$

The proposed procedure builds a binary representation of the sentence typed-dependency graph that is used for fast sub-linear lookup of similar sentences in the sample database. General type kernels return valuable output for similar and dissimilar instances, but tree kernel functions for relation extraction [2] return a normalized, symmetric similarity score in range [0,1]. Dissimilar trees have zero score. Therefore, we can optimize tree kernel evaluation process by limiting it to the similar instances only.

It can be the simple threshold strategy:

$$\{s : (\mathbf{V}_s, \mathbf{V}_p) > \theta\}, \quad (5)$$

where $\mathbf{V}_s$ - sample vector, $\mathbf{V}_p$ - probe vector, θ - threshold.

Or the adaptive threshold strategy:

$$\{s : \frac{(\mathbf{V}_{s^*}, \mathbf{V}_p) - (\mathbf{V}_s, \mathbf{V}_p)}{(\mathbf{V}_s, \mathbf{V}_p)} > \theta\}, \quad (6)$$

where $\mathbf{V}_{s^*}$ - sample with the highest score.

2.5. Matching typed-dependency graph fragments

There are two kinds of the node similarity in typed-dependency graphs:

- The similarity of the dependant sub-graphs (top-down similarity), which is discussed in the previous subsection.
- The similarity of the governing sub-graphs (bottom-up similarity), which haven't discussed yet.

A simple way to incorporate the bottom-up similarity is to build the disjunction of the node representation and its ancestor role representations (see Figure 2 and [5] for details). However, this approach works only for simple cases, because it treats all ancestor roles equally and ignores ancestor representations.

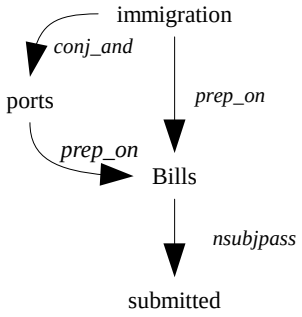


Figure 2: Node roles

The modification of steps 4 and 5 of the procedure from the previous subsection allows to build better representations for the mapping, but at the cost of higher complexity.

4. Use CDT procedure to build a representation of the reverse dependency relation:

$$\mathbf{V}^{dependency}_{m_{gov}, w_{dep}} = \left\langle \begin{array}{l} \left\langle \mathbf{V}^{gov}_{dependency} \vee \mathbf{V}_{m_{gov}} \right\rangle \\ \vee \\ \left\langle \mathbf{V}^{dep}_{dependency} \vee \mathbf{V}_{w_{dep}} \right\rangle \end{array} \right\rangle \quad (5)$$

where $\mathbf{V}_{m_{gov}}$ - random vector of the governing node, $\mathbf{V}_{w_{dep}}$ - CDT vector of the dependent word. E.g,

$$\mathbf{V}^{agent}_{m(submitted), w(Brownback)} = \left\langle \begin{array}{l} \left\langle \mathbf{V}^{gov}_{agent} \vee \mathbf{V}_{m(submitted)} \right\rangle \\ \vee \\ \left\langle \mathbf{V}^{dep}_{agent} \vee \mathbf{V}_{w(Brownback)} \right\rangle \end{array} \right\rangle$$

5. Use disjunction of reverse dependency relation representations to build a representation of the dependent node. E.g.,

$$\mathbf{V}_{m(Brownback)} = \mathbf{V}^{nsubjpass}_{m(submitted), w(Brownback)}$$

Aggregate similarity of graph fragments is a linear combination of the top-down and bottom-up similarities:

$$sim(x, y) = \alpha(\mathbf{V}_{n(x)}, \mathbf{V}_{n(y)}) + \beta(\mathbf{V}_{m(x)}, \mathbf{V}_{m(y)}) \quad (6)$$

where $\mathbf{V}_{n(x)}$ and $\mathbf{V}_{n(y)}$ - top-down binary representations of nodes *x* and *y*, $\mathbf{V}_{m(x)}$ and $\mathbf{V}_{m(y)}$ - top-down binary representations of nodes *x* and *y*, α and β - weighting factors.

2.6. Fact extraction

Facts are defined as entity relations of the sentence. All sentences from the sample database contain typed-dependency tree augmented with relation edges. Probe doesn't have relation edges. Procedure of the fact extraction is the following:

1. A sample with maximum similarity with the probe is extracted from the database:

$$s^* = \arg\max_{s \in S}(\mathbf{V}_{n(s)}, \mathbf{V}_{n(p)})$$

where *S* – samples database, $\mathbf{V}_{n(s)}$ - sample top-down node vector, $\mathbf{V}_{n(p)}$ - probe top-down node vector. Top nodes of *p* and *s* added to match node set {(*p*,*s*)}

2. Child nodes of the current matched nodes are greedy matched by maximum similarity and added to match node set.
3. Relation edges between nodes in the sample transferred to matched nodes in the probe.

3. Related work

There are models and methods of distributed representation for encoding data structures in vectors, matrices, or high-order tensors. However, vector representation is the most promising from the computational point of view (see [4]).

Distributed Tree Kernels [14-15] use the following definition of the distributed vector representing the subtrees of tree *T*:

$$\widetilde{\mathbf{T}} = \sum_{n \notin N(T)} s(n), \quad (7)$$

where $N(T)$ – set of nodes of the tree T, n – node, $s(n)$ – sum of the distributed vectors of the subtrees of T rooted in node n. The function $s(n)$ is recursively defined as follows:

- $s(n) = \mathbf{n} \otimes \mathbf{w}$ if n is a pre-terminal node $n \to w$, where $\mathbf{n}$ – vector representing the node n, $\mathbf{w}$ – vector representing the word w.
- $s(n) = \mathbf{n} \otimes (\mathbf{c}_1 + s(c_1)) \otimes \ldots \otimes (\mathbf{c}_n + s(c_n))$ if n is not a pre-terminal node $n \to c_1, \ldots, c_n$, where $\mathbf{c}_1, \ldots, \mathbf{c}_n$ - vectors of the child nodes $c_1, \ldots, c_n$.

The binding function $\otimes$ is either the reverse element-wise product $\mathbf{v} = \mathbf{a} \otimes \mathbf{b}$:

$$v_i = \gamma a_i b_{n-i+1}, \quad (8)$$

(where v_i, a_i and b_{n-i+1} - elements of the corresponding vectors, γ - normalization factor), or discrete vector convolution [13].

Two types of Distributed Tree Kernels are proposed [14-15]:

- Pure Distributed Tree Kernels that use random vectors with elements drawn independently from a normal distribution $N(0,1)$.
- Distributional Distributed Tree Kernels that use LSA word vectors obtained from a large text corpus.

To reduce the computational complexity of distributed tree kernels, an embedding scheme is proposed [15].

4. Conclusions

Binary distributed representation of structured data allows easy estimation of the complex object similarity and effective parallel implementation. However, a new representation and matching scheme for any new problem needs to be developed.

Binary distributed representation scheme for typed-dependency representation is proposed, and retrieval and matching procedures are outlined. Experimental evaluation showed that that using of binary distributed representations for fact extraction is possible, but random word vectors fail to capture the word similarity and distributional binary representations of the word vectors need to be developed in order to incorporate distributional word semantic. Distributional Distributed Tree Kernels use LSA vectors to capture the word similarity, but LSA real-valued vectors do not apply to the binary distributed representation schema. Experimental evaluation of the randomized projective methods for construction of binary sparse vector representations [16] is needed.

5. References

[1] Radinsky K. and Horvitz E., "Mining the Web to Predict Future Events", *Proc. of International Conference on Web Search and Data Mining*, 255-264, 2013.

[2] Culotta A. and Sorensen J., "Dependency Tree Kernels for Relation Extraction", *Proc. of the 42nd Annual Meeting on Association for Computational Linguistic*s, 2004.

[3] Poon H. and Vanderwende L. "Joint Inference for Knowledge Extraction from Biomedical Literature", *Proc. of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, 813-821, 2009.

[4] Rachkovskij D.A. and Slipchenko S. V., "Similarity-Based Retrieval with Structure Sensitive Sparse Binary Distributed Representations", *Computational Intelligence*, 28(1), 106-129, 2012.

[5] Slipchenko S.V. and Rachkovskij D.A., "Analogical mapping using similarity of binary distributed representations", *Information Theories and Applications*, 16(3), 269-290, 2009.

[6] Slipchenko S.V. and Rachkovskij D.A., "Mapping and inference by analogy based on neural distributed representations", *Proc. of Knowledge-Dialogue Conference*, 9, 95-101, 2009.

[7] Forbus K.D., Gentner D. and Law K., "MAC/FAC: A model of similarity-based retrieval", *Cognitive Science*, 19(2), 141-205, 1995.

[8] Falkenhainer B., Forbus K.D. and Gentner D., "The Structure-Mapping Engine: Algorithm and Examples", *Artificial Intelligence*, 41, 1-63, 1989.

[9] De Marneffe M.-C. and Manning C.D., "The Stanford typed dependencies representation", *Proc. of COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*, 1-8, 2008.

[10] Wilson R.C., Hancock E.R. and Luo B., "Pattern vectors from algebraic graph theory", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7), 1112-1124, 2005.

[11] Riesen K., Neuhaus M. and Bunke H., "Graph embedding in vector spaces by means of prototype selection", *Graph-Based Representations in Pattern Recognition (Lecture Notes in Computer Science)*, 4538, 383-393, 2007.

[12] Rachkovskii D. A.,Slipchenko S. V., Kussul' E. M. and Baidyk T. N., "A Binding Procedure for Distributed Binary Data Representations", *Cybernetics and Systems Analysis*, 41(3), 319-331, 2005.

[13] Plate T., Holographic Reduced Reprsentation, 2003.

[14] Zanzotto F.M. and Dell'Arciprete L., Distributed structures and distributional meaning, *Proc. of the Workshop on Distributional Semantics and Compositionality*, 10–15, 2011.

[15] Zanzotto F.M. and Dell'Arciprete L., Distributed Tree Kernels, *Proc. of International Conference on Machine Learning*, 193-200, 2012.

[16] Rachkovskij D.A., Misuno I.S. and Slipchenko S.V. Randomized projective methods for construction of binary sparse vector representations, *Cybernetics and Systems Analysis*, 48(1), 146-156, 2012.