Selecting On-Topic Sentences from Natural Language Corpora

Michael Levit[1], Elizabeth Boschee, Marjorie Freedman

BBN Technologies, 10 Moulton st, Cambridge, MA

{mlevit,eboschee,mfreedma}@bbn.com

Abstract

We describe a system that examines input sentences with respect to arbitrary topics formulated as natural language expressions. It extracts predicate-argument structures from text intervals and links them into semantically organized *proposition trees*. By instantiating trees constructed for topic descriptions in trees representing input sentences or parts thereof, we are able to assess degree of "topicality" for each sentence. The presented strategy was used in the BBN distillation system for the GALE Year 1 evaluation and achieved outstanding results compared to other systems and human participants.

Index Terms: machine learning, question answering, topicality.

1. Introduction

One of the best studied problems of machine learning is topic classification. Its applications to natural language processing are numerous and can be viewed as multiclass multi-label classification tasks [1]. While topic classification can deal with fairly large number of topics, this number remains finite and the topics must be known in advance. However, if these preconditions are not satisfied, and topics are allowed to be specified at run-time via natural language formulations, the classification paradigm can not be straightforwardly applied anymore and other solutions must be searched for.

We formulate the *topicality* problem as this: given a free-text formulation of a topic and a natural language document, determine the extent to which the document is relevant for the topic. It is not difficult to imagine how knowledge of this type could be beneficial for several application domains, such as mixed-initiative dialog systems and story segmentation, but we see its biggest advantage for the question answering task and, in particular, for the kinds of queries that either are focused entirely on a topic or event (e.g. *"List facts about events described as follows:* [EVENT]*"*) or inquire about a particular aspect of a topic or the reaction an event elicited (e.g. *"How did* [COUNTRY] *react to* [EVENT]*?"*). In the example queries above, [EVENT] is a *slot* filled with a free-text description of some event. To select passages from a corpus that answer such queries we need to be able to say how "on topic" they are with respect to such free-text formulations, that is, we need to solve the topicality problem.

Matching a slot can be trivial if the slot consists of a single name or very specific term, but it can also require considerable effort when the slot contains a lengthy or complex description of some topic or event. One example of a difficult case is the following query from the GALE Distillation task [2]: *"List facts about* [the looting of Iraqi museums after the U.S. invasion]*"*. Another difficulty in dealing with queries that have topic and event slots comes from the fact that human language makes it possible to convey on-topic information without using the words that were used in the actual topic formulation: *"Many works of art were stolen from Baghdad galleries in 2003"*. Neither does the presence of the topic words in a sentence guarantee that it will be relevant: *"There were no known instances of looting of Iraqi museums before the U.S. invasion"*. The topic-based GALE Year 1 templates frequently posed such problems. In this paper we explain how free-text query slots can be represented as predicate-argument "proposition trees", and how responses to queries involving free-text arguments can be found using a matching algorithm to approximately instantiate the query proposition tree in the proposition trees of the candidate responses.

2. Related Work

One example of semantic predicate-argument structures are *dependency relations*. In [3] it was shown how to construct hierarchies of typed dependency relations using a collection of handcrafted rules. Each relation connects a semantic head with its dependent. Relations connect individual words, and the set of relation types is also hierarchically organized but not lexicalized. These are the two major differences between dependency relations and our predicate-argument structures, where text intervals spanning entire *mentions* are connected and some lexical items (e.g. prepositions) give rise to separate argument types.

Predicate-argument structures have been previously used to answer questions with difficult semantics. Moldovan at al. [4] suggested transforming question and answer sentences into logical representations and using first order logic to produce an inference between them. They used world knowledge and NLP axioms to facilitate derivation. Logical representation of a sentence was built upon a number of semantic predicates extracted from its syntactic parse.

In [5] one piece of evidence used to determine the expected semantic role of a potential answer in a sentence came from PROPBANK predicate-argument structures [6] extracted from unstructured questions, and in [7] hierarchies of lexical, syntactic and semantic sentence representations (of which predicate-argument structures were a part) were employed to decide if one sentence *entailed* (implied) another. A number of transformations such as genitive replacement (*"Z's W"* is equivalent to *"W of Z"*) was incrementally applied in order to match structures using unification. Semantic role labeling of relation arguments was trained on PROPBANK and, in the reported experiments, restricted to verb frames only.

3. Propositions and Proposition Trees

We used BBN's state-of-the-art information extraction system SERIF [8] that automatically extracts predicate-argument struc-

[1]Michael Levit is currently affiliated with International Computer Science Institute, Berkeley, CA

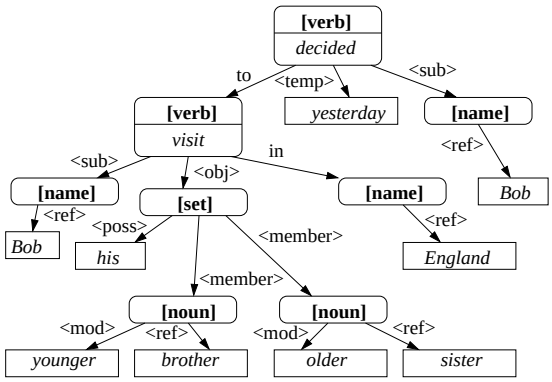

Figure 1: Proposition tree for sentence *"Yesterday, Bob decided to visit his younger brother and older sister in England"*.

tures (called *propositions*) from syntactic parses. Each proposition has a type (e.g. VERB, NOUN, MODIFIER, SET and others), a head predicate and a non-zero number of arguments. Each argument refers to another proposition or a mention (associated with a subtree of the sentence's syntactic parse). Each argument also has a role, where the lexicon of roles consists of a closed set of grammatical functions such as subject, object, indirect object and others, as well as a "lexicalized" set of preposition- and conjunction-based roles (e.g. *to*, *of*, *that*). Mention arguments are by far the most common type of argument. Consider the sentence: *"John throws the ball"*. The following three propositions can be extracted from it:
p_1 = throws <verb> (<sub>: m_1, <obj>: m_2)
p_2 =<name> (<ref>: m_1, "John")
p_3 = ball <noun> (<ref>: m_2)

The first proposition is of type VERB and has head predicate *throws*. Two arguments, both mentions, are attached to it: the first one referring to *John* as the subject of the proposition and the second one to *the ball* as its object. Furthermore, two propositions (a NAME proposition and a NOUN proposition) are created to represent each of these entities.

Because of the way propositions are extracted from syntactic parse trees, it should be possible to assemble the individual propositions extracted from a sentence into one coherent tree structure representing the semantic interpretation of the sentence, abstracted from its exact syntactic realization. In practice, a sentence may consist of either one or several such *proposition trees*, created by replacing mention arguments of propositions with propositions referencing those mentions as their predicate head. We use the notation *proposition forest* to refer to all proposition trees extracted for a sentence (or any other interval of text). Figure 1 shows a proposition tree that represents this sentence: *"Yesterday, Bob decided to visit his younger brother and older sister in England"*. In this diagram, box labels in bold indicate proposition types and labels in italics signify words or word phrases associated with each tree node. Arc labels are argument roles. Note that because of the way proposition trees are built, it doesn't matter if verbs are used in active or passive form, so *"John throws the ball"* and *"The ball is thrown by John"* would result in the same proposition tree.

In practice, the basic strategy for creating proposition trees from sentences is this:

1. for each proposition extracted from the sentence, create a proposition tree rooted in that proposition, then recursively expand the tree by replacing its arguments by appropriate propositions, where possible;
2. if the resulting tree is already subsumed by another, bigger proposition tree, ignore it;
3. use appropriate criteria to select one proposition tree to represent the sentence, or return the entire proposition forest.

Note that proposition trees can also be extracted from spans of text that are not well-formed sentences. By analyzing all propositions contained within any text interval, we can represent that interval as a proposition forest. In particular, this holds for the query slots as described above, which are almost always sentence fragments.

4. Topic Matching with Proposition Trees

In order to estimate the extent to which a sentence is "on topic" with respect to a free-text slot, we extract proposition forests from both the slot and the sentence. We then try to instantiate each of the slot proposition trees τ_i in each of the sentence proposition trees t_j. If an instantiation is possible, its score θ_{ij} is compared with other instantiation scores for this slot proposition tree, and the non-negative maximum is saved:

$$\theta_i := \max(\max_j \theta_{ij}, 0). \quad (1)$$

Finally, the *topicality measure* is defined as:

$$\theta := \frac{\sum_i w_i * (\theta_i / \theta_i^{max})}{\sum_i w_i}. \quad (2)$$

Here w_i is the weight of the proposition tree τ_i (usually reflecting its number of nodes) and θ_i^{max} is its pre-computed highest possible instantiation score.

4.1. Exact Instantiation of Proposition Trees

Next we describe how one proposition tree (pattern tree) can be instantiated in another (subject tree). We are only interested in tree containment: the nodes and links of the pattern must be present in the subject, but not vice versa. This problem is equivalent to the classical subtree matching problem [9], and can therefore be solved in time linear in pattern and subject size. Because in practice the trees we consider are rather small, we are not concerned with theoretical complexity, and therefore adopt a modification of a depth-first recursive matching algorithm with optimistic cost estimation and other lossless heuristics.

In the exact match formulation, two nodes are considered a match for each other if they are labeled with the same word (or phrase), and if they can be reached from their parents over links with identical roles, whereby the parent nodes themselves have already been found to be a match. The tree matching score θ_{ij} in the exact case is either 0 or 1. Figure 2 shows a pattern tree (a) and a subject tree (b) in which it can be instantiated exactly.

4.2. Extensions

If we allow only exact matches in our instantiation scheme, we will miss most of the relevant sentences. For instance, none of the following trivial sentences would be identified as relevant for the pattern expression in Figure 2: *"the plane flew"*, *"the plane goes to England"*, *"plane's flight"*, *"it* [the plane] *takes off and flies"*.

To handle these kinds of variations, we developed several extensions to the exact instantiation scheme. To support them,

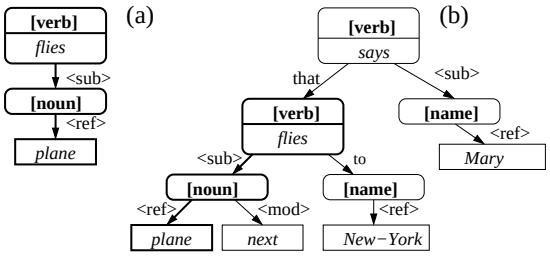

Figure 2: Proposition tree created for expression *"The plane flies"* (a) can be instantiated in the proposition tree for *"Mary says that the next plane flies to New-York"* (b).

we need to allow not one but possibly many words or word phrases to be associated with each tree node, forming *word pools*. Then, the word matching condition for two nodes is relaxed so that two nodes match if the intersection of their word pools is not the empty set. Furthermore, the tree instantiation score θ_{ij}, formerly a binary function, is now computed according to:

$$\theta_{ij} = \#\text{matched} * \gamma^{\text{match}} - \gamma^{-} \quad (3)$$

where γ^{match} is a constant score awarded to each node-to-node match, $\#\text{matched}$ is the number of such matches in a tree instantiation and γ^{-} is the cumulative cost of these matches. There are several factors that influence the cost of a match. First, each word in a word pool can have a cost associated with it, so that if several matching word pairs are possible, we would select the one with the lowest sum of word costs. We will also in some instances relax the constraint that the matching nodes must be reached via links with identical roles, and this relaxation will incur additional cost.

We now describe matching extensions in detail.

4.2.1. Mentions of the Same Entity

Before proposition trees are created from a sentence, SERIF extracts ACE-entities from it [10]. If a node of a proposition tree can be associated with a mention that belongs to an entity, then all mentions of this entity (even if they occur in other sentences) can contribute to the word pool of the node[2]. Thus, even if a sentence says *"it flies"*, but *it* and *the plane* are mentions of the same ACE-entity, then *plane* will be added to the word pool of the tree node containing *it*.

4.2.2. WordNet Stemming and Synonyms

For each word in a word pool of a tree node, we use WORDNET [11] to add its stemmed version as well as synonyms and directly connected hypernyms and hyponyms to the pool. The cost of the new words is computed as the cost of the original word incremented by empirically estimated stemming and/or synonym costs respectively. This extension will allow us to instantiate *"the plane flies"* in *"the plane goes to England"*.

4.2.3. External Dictionaries

In a similar manner we can use other sources to expand word pools of tree nodes. For instance, we can add trained alternative spellings for names or use gazetteers to add nationalities and capitals to country names. All of the additions must be accompanied with appropriate costs.

[2]In the present implementation, pre-terminal syntactic heads of the mentions are used for this purpose.

4.2.4. Role Mismatch and Other Structural Transformations

Earlier, we explained how synonyms can be added to node word pools. Along the same lines of thought, we argue that some argument roles are more similar than others. For example, proposition trees for *"Abu Abbas's arrest"* and *"the arrest of Abu Abbas"* differ only because the link between *arrest* and *Abu Abbas* is labeled *<possessive>* in the first case and *of* in the second. We can allow for certain role confusions to take place while penalizing them with appropriate costs.

This extension is particularly relevant for the distillation task where query writers often choose to express topics or events as nominalizations, but candidate responses might describe the same events in their verbal forms. For instance, the passage *"Bush talked to Blair"* is a perfect match for the event slot *"Bush's communications with Blair"*. In order to account for this transformation, relaxed argument role matching is necessary. We also allow additional word pool matching between verbs and their nominalizations (e.g. *communication* for *communicate*, which will subsequently evoke synonym *talk*).

Finally, if there is a set-proposition on the subject tree side (*"Bush went to England and talked to Blair"*), one should be able to "fall through" down the link to one of its members. This operation incurs a small penalty.

4.2.5. Missing Arguments

While it is always preferable to instantiate all nodes of a pattern proposition tree, many relevant sentences will contain only a partial match. It will be highly unlikely to find many sentences where we can fully instantiate the proposition tree for *"the looting of Iraqi museums after the U.S. invasion"*. On the other hand, the subtree *"the looting of Iraqi museums"* has a much better chance of being fully instantiated (especially with the above extensions to the matching algorithm). Thus, we might allow a pattern tree to "match" a subject tree even if not all of its subtrees can be instantiated therein, as long as a significant portion of the pattern tree is in fact instantiated with a sufficiently high score. The incurred penalty in this case is proportional to the size of the non-instantiated subtree of the pattern tree.

4.3. Name-Aware Instantiation

Some instantiation mismatches should be considered more serious than others. For instance, if a topic description contains a name, this name should be in the word pool of at least one node of the proposition forest of the sentence (or its neighbors). This way we significantly reduce the risk of finding *"the looting of Afghani museums after the U.S. invasion"* when asked about looting in Iraq. We call this strategy *name-aware proposition tree instantiation*. Our experiments showed that it dramatically improved the precision of found passages.

5. Back-off Strategies

A non-negative topicality measure θ can only come about if there is at least some significant structural similarity between the pattern proposition tree and some proposition (sub)trees of the subject. However, humans can express the same information in ways that share virtually no semantic structure. It is conceivable (in fact very likely) that most of the "good" answers to the looting query will not have exactly the same structure as the slot itself. For instance, *"In 2001 U.S. troops invaded Iraq, and subsequently many Iraqi museums were looted"*. While both facts (invasion and looting) are present in the sentence, they are

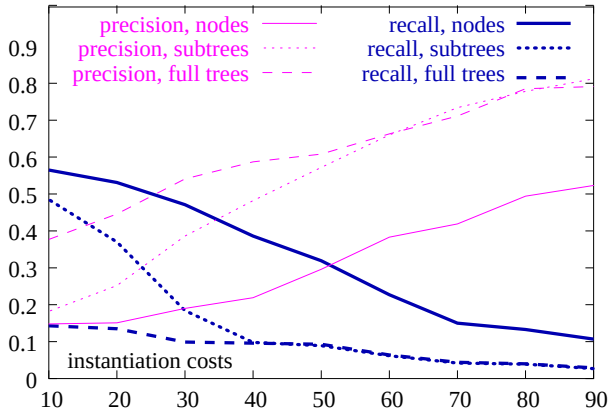

Figure 3: Dependency of precision and recall of three topicality measures on costs.

not connected in the same way they are connected in the slot proposition tree.

We decided to introduce a modified topicality measure η based on subtrees of (full) proposition trees. While computing subtrees we allow for only a minimum amount of proposition expansion (see Section 3), so that each slot is represented by a number of such proposition subtrees (for instance, in the "looting" query we'd build one subtree for "the U.S. invasion" and one for "looted museums"). The rest of the computation scheme remains the same as presented in eqs. (3), (1) and (2), with the only exception that the weights w_i of the proposition subtrees in eq. (2) also depend on the size of the original full trees that contain them. A subject will have high relevance if many of the subtrees of the query slot are instantiated in it, even if these subtrees are not connected to each other.

Finally, there is yet another even more generous topicality measure μ that indicates the percentage of tree nodes in the slot forest whose word pools have at least one match in the word pools of the tree nodes of the sentence forest. This is analogous to an advanced "bag-of-words" matching strategy that treats tree nodes rather than words, and incorporates the word pool cost structure of the proposition trees to allow for more sophisticated matching.

In practice, we say that a sentence is relevant on the given topic if one of the three measures (θ,η or μ) exceeds its respective, empirically determined threshold.

6. Results

Quite intuitively, all three topicality measures complement each other in terms of their precision/recall characteristics. While topicality with full trees is especially useful to produce high precision answers, the tree node topicality operates in a high recall area, and subtree topicality lies between the two. Figure 3 illustrates this fact using first query template (*"List facts about events described as follows:* [EVENT]*"*) of GALE Year 1 evaluation. It shows how the fractions of n-grams in reference/found answer snippets that could be instantiated in found/reference snippets (measures that we associate with recall and precision) depend on cut-off for instantiation cost threshold. Our GALE Year 1 evaluation system operated downstream of a TF-IDF document retrieval engine and used a combination of all three of the topicality measures. For the template above it achieved information content F-measure of 0.46, about 3 times higher than other systems and 1.5 times higher than an average human (restrained in time). Our system was also clearly the highest ranking in terms of *proficiency*, an information-theoretical measure defined as a mutual information normalized to the [0,1] interval [2]. For the other two query templates relying primarily on topicality (*"Find statements made or attributed to* [PERSON] *on* [TOPIC]*"* and *"How did* [COUNTRY] *react to* [EVENT]*?"*) we achieved information content F-measures of 0.42 and 0.23 respectively, almost twice as high as the closest competitor system.

7. Conclusion

We described a system that answers the "on-topic" question for natural language documents and arbitrary free-text topic descriptions. The approach is based on creating proposition trees, semantically organized hierarchies of predicate-argument structures, for topic formulation and document sentences and an error-tolerant tree instantiation mechanism. We successfully used the approach in BBN's distillation system for GALE Year 1 evaluation where it achieved outstanding extraction results.

8. References

[1] Levit, M.: *"Spoken Language Understanding without Transcriptions in a Call Canter Scenario"*; Ph.D. thesis, Logos Verlag, Berlin, Germany, 2005.

[2] BAE Systems: *"Go/No-Go Formal Distillation Evaluation Plan For GALE"*, 2006, URL: https://www.sainc.com/GALE-BAA/public/.

[3] M.-C. de Marneffe, B. MacCartney and C. D. Manning: *"Generating Typed Dependency Parses from Phrase Structure Parses"*; in Proc. of LREC, Genoa, Italy, 2006.

[4] D. Moldovan, C. Clark and S. Harabagiu: *"COGEX: A Logic Prover for Question Answering"*; in Proc. of HLT-NAACL, Edmonton, Canada, 2003.

[5] S. Narayanan and S. Harabagiu: *"Question Answering Based on Semantic Structures"*; in Proc. of the 20th. Int. Conf. on Computational Linguistics (COLING 2004), Morgan Kaufmann, San Francisco, CA, 2004.

[6] P. Kingsbury, M. Palmer and M. Marcus: *"Adding Semantic Annotation to the Penn TreeBank"*; in Proc. of the Human Language Technology Conference (HLT'02), San Diego, CA, 2002.

[7] R.de Salvo Braz, R. Girju., V. Punyakanok, D. Roth and M. Sammons: *"Knowledge Representation for Semantic Entailment and Question Answering"*; in Proc. of IJCAI-05 Workshop on Knowledge and Reasoning for Question Answering.

[8] L. Ramshaw, E. Boschee, S. Bratus, S. Miller, R. Stone, R. Weischedel and A. Zamanian: *"Experiments in Multi-Modal Automatic Content Extraction"*; in Proc. of HLT-01, San Diego, CA, 2001.

[9] C. M. Hoffmann and M. J. O'Donnell: *"Pattern Matching in Trees"*; in Journal of the ACM, 29(1), pp.68–95, 1982.

[10] LDC: *"ACE English Annotation Guidelines for Entities"*; 2005, URL: http://www.ldc.upenn.edu/Projects/ACE/.

[11] C. Fellbaum: *"WordNet, an Electronic Lexical Database"*; MIT Press, 1998.