2019 IEEE 13th International Conference on Semantic Computing (ICSC)

Siamese Networks for Semantic Pattern Similarity

Yassine Benajiba*, Jin Sun†, Yong Zhang†, Longquan Jiang†, Zhiliang Weng† and Or Biran‡
*Symanto, †Mainiway, ‡Elemental Cognition
*yassine.benajiba@symanto.net, ‡orb@elementalcognition.com
†{jin.sun,yong.zhang,longquan.jiang,zhiliang.weng}@mainiway.com

***Abstract*—Semantic Pattern Similarity is an interesting, though not often encountered NLP task where two sentences are compared not by their specific meaning, but by their more abstract semantic pattern (e.g., preposition or frame). We utilize Siamese Networks to model this task, and show its usefulness in determining SQL patterns for unseen questions in a database-backed question answering scenario. Our approach achieves high accuracy and contains a built-in proxy for confidence, which can be used to keep precision arbitrarily high.**

I. Introduction

The *Semantic Textual Similarity* (STS) task [1], which places pairs of sentences on a scale ranging from completely unrelated to perfect paraphrases (with intermediates such as topic similarity, partial coverage and near-identity), has gained much attention in recent years. In addition to being interesting on its own as a step towards semantic modeling of sentences, STS is an important (explicit or implicit) sub-task for many NLP applications such as Machine Translation, Generation, Summarization and Question Answering. Sentence-level similarity is significantly harder than word- or term-level similarity because of the variant length and semantic complexity of sentences, which makes it harder to model with traditional approaches (e.g., bag-of-words, tf-idf and similar models). Approaches to STS include pooling of word-level embeddings [2], fixed-length encodings of sentences [3]–[5], matrix factorization approaches [6], [7] and neural similarity models [8], [9].

We explore the similar but less frequently encountered task of *Semantic Pattern Similarity* (SPS). Similar to STS, this task brings about its own challenges as a semantic task and is potentially useful for NLP applications. Semantic patterns are conceptually templates in which certain types of arguments are expected, and which model some abstraction of the semantics of a text unit - often an abstraction of entities, attributes and relations. One common type of semantic patterns includes linguistic semantic representations such as prepositions, frames and AMR [10], usually encountered in the context of the Semantic Role Labeling or Semantic Parsing tasks; however, semantic patterns may be formed with other structured representations of text, which result in a different kind of abstraction.

Like STS, in SPS we are given two sentences. Instead of modeling the similarity of the sentences, however, we want to model only the similarity of the underlying semantic patterns. For example, the sentences "John sold a car to Mary in 2016" and "Bob bought an apple from Jane on Sunday" have identical semantic patterns. The model must therefore learn to distinguish between text relevant to the pattern and text relevant only to the specific sentence.

In this paper, we use SQL queries as proxies for semantic patterns. We use the recently introduced WikiSQL data set [11], which contains over $87,000$ natural language questions aligned with SQL queries that produce the answer (from a known database). The benefits of using this data set are threefold: first, it is unusually large for this sort of annotated aligned data set; second, it is relatively straightforward to define a non-binary distance metric between SQL templates; and third, succeeding in the SPS task for this data set represents a large step towards the important real-world application of SQL-backed question answering. To make this third point more concrete, we evaluate our SPS approach by attempting to find, for an unseen single question, a question in the training data which shares a SQL template with it. In other words, we use our SPS model for the more challenging task of finding the semantic pattern (SQL template, in this case) of a new question. As an example of the usefulness of this task, it can be used as a sub-task in a SQL-backed QA scenario: first find the SQL template for a question; then fill in the blanks using named entities from the question or synonyms thereof, a task analogous to semantic parsing.

To model the similarity of two questions, we use recurrent Siamese neural networks, an architecture that was successfully used for the closely related STS task [9]. Our model achieves a high accuracy, and we show that the predicted similarity acts as a confidence score, so that thresholding it can keep precision arbitrarily high at the cost of not handling all cases.

II. Related Work

While SPS, to the best of our knowledge, has not explicitly been pursued as a task, it is alluded to in the literature of related tasks. In addition to work on the STS task, which is briefly surveyed in the previous section, some relevent work exists in the context of paraphrasing. [12] mine *paraphrasal templates* - groups of concrete textual templates which would be paraphrases if filled with the same entities - from Wikipedia. Their approach relies on first finding and removing entities, and then clustering the remaining templates in a lexical vector space. In contrast, we model sentences directly in semantic pattern space. Earlier examples from the

* ‡ Work done in collaboration with Mainiway AI Lab

978-1-5386-6783-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSC.2019.00044

paraphrasing literature include [13], which uses heuristic rules to find short templated paraphrases, and [14] who produce *slotted lattices* from a comparable corpus which contains paraphrases.

In Natural Language Generation, [15] used similar grouped templates, but did not use a similarity metric (instead relying only on entity types to group templates). [16] extract semantic templates from Wikipedia pages by aligning them with entities from Semantic Web data. [17] and [18] generate novel sentences by performing edits on another sentence, which tend to preserve the original semantic pattern. In both cases, the semantic pattern similarity is not modeled directly, but emerges from the model's learned behavior of making certain types of small lexical edits.

While not as directly related, many standard practices in NLP apply some form of abstract text matching. For example, in Machine Translation, *alignment templates* [19] can be thought of as a kind of semantic pattern matching across languages, while in Information Extraction, Hearst Patterns [20] and similar techniques where a set of lexicalizations of a single relationship are used to mine pairs of words or entities for which that relationship holds can be said to apply semantic pattern matching to these pairs. None of these approaches provide a semantic similarity score for arbitrary sentences.

III. Data

We use the WikiSQL data set [11], which contains aligned pairs of questions and SQL queries over database tables collected from Wikipedia. As described in the paper, the questions and queries are created with a hybrid algorithmic/templated approach and human editors (on Amazon Mechanical Turk) who perform the final matching and filtering.

WikiSQL is the largest aligned text-SQL dataset publicly available. The queries are limited to having one select column, no nested queries, and no joins, which may limit its usefulness for real-world applications, but is an attractive property for research purposes as it constrains the problem domain.

The data is fairly noisy: some questions are incoherent, and some do not match their paired SQL query. The column types of the database tables are not always correct, and most are mistakenly labeled as "text" by default. To alleviate these problems, we did some cleaning of the data. First, we used regular expressions to find numeric and date columns and correctly assign their types; and second, we removed questions that were too short or contained few alphabetic characters. We will release the resulting adjusted data set with a detailed description of the adjustments upon acceptance.

IV. Our Approach

Given an unseen question, our goal is to find, in a pool of questions, another question with the same semantic pattern (SQL template).

We employ a Siamese LSTM regression model to predict the similarity of the SQL templates of two questions (Section IV-B). Instead of comparing the unseen question to the entire training set, which would be costly, we cluster the training set ahead of time using a lexical representation of the questions (Section IV-C), and compare a new question only to the members of its nearest cluster.

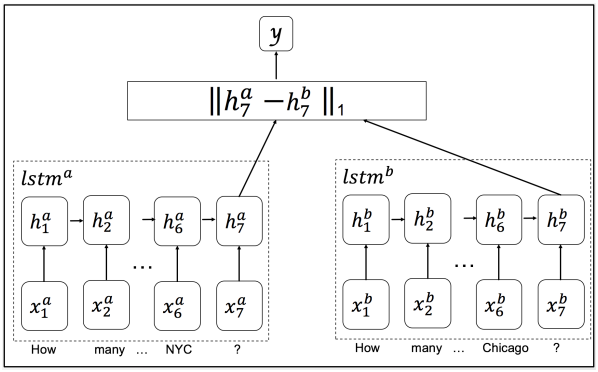

Fig. 1. Siamese LSTM Architecture

A. SQL Structure Distance

We define the SQL structure distance as:

$$SQLSD(sql_i, sql_j) = \sum_{c \in C} I(sql_i(c), sql_j(c))$$

where I is an indicator function that equals 1 when its two arguments match. $sql_i(c)$ is a function that indicates the value of the SQL query sql_i for the SQL constituent c. Finally, C is the set of constituents we take into consideration to compute the distance between two queries. It consists of the *type* of the SELECT column (text, number, date); the *aggregator* of the selection (none, COUNT, SUM, etc.); and the number of elements in the WHERE clause for each condition ($=$, $>$, $<$, etc.). These are enough because of the limited complexity of the data set, as described in Section III.

B. Siamese LSTM Regressor

Siamese networks, designed as a way to compare two objects, were first introduced by [21]. By focusing on comparison, the network is not tied to a particular set of classes. A Siamese network can be diagrammed as two branches (although both branches share the same parameters), both feeding to one layer that performs some distance measure to get an output. The idea behind this architecture is to learn a transformation function that transforms the input representation into a space where similar objects have similar representations.

[9] use Siamese networks to compare pairs of sentences. Unlike [21], they use a recurrent network to learn the transformation function as it is more appropriate for modeling language. We use a similar architecture, shown in Figure 1. The major difference is that given the nature of our distance metric, we train our network to learn a regression function. The loss function is the mean squared error from the distance measure $SQLSD$.

C. Question Lexical Clustering

Comparing an unseen question to every question in the training set would be prohibitively (and unnecessarily) costly for a real-world QA scenario. To reduce the size of our search space, we first cluster the training set instances using a one-hot lexical representation of the questions, using as a vocabulary all words appearing in the training set with a frequency $> \alpha$ (in our experiments, we set $\alpha = 50$. This number was found empirically by tuning on the dev set). We then use k-means to cluster the data in this vector space. With $k = 500$, we obtain an average cluster size of 122.6.

The one-hot representation is most adequate for the problem at hand because the questions are typically short, and the frequency of each term is almost always 1. In addition, because we are interested in the semantic pattern rather than the semantics themselves, functional words are important; word embeddings or other distributional representations may not discriminate between them.

Because of the choice of α, the questions are clustered mostly by these functional words; as a result, questions in different clusters tend to be questions with different functional words (e.g., "how" and "many" vs. "who" and "was"). The clustering therefore maximizes the inter-cluster SQL structure distance: while two questions in the same cluster are not guaranteed to have the same SQL template, questions in different clusters are much *less* likely to share a template, and therefore it is relatively safe to ignore them and focus only on the most similar cluster at prediction time.

V. Experiments and Results

In our experimental set up, for each question in the test (or dev) set, we try to find a question in the training set which has the exact same SQL template, and evaluate our success using the (binary) accuracy of the selection.

One advantage of using a soft regression score for a binary task is being able to use a minimum threshold. In a real-world application, it is often useful to keep a very high accuracy for accepted instances, even at the cost of rejecting some instances ("unable to handle" is a better response than a wrong answer). To that end, we define a threshold β, such that if there is no training question in the cluster for which the network predicts a distance $< \beta$ from the test question, we simply reject the test question. This results in a "safe classification" as defined by [22].

In our evaluation, we therefore track two quantities for varying values of β (from 0.1 to 2.5): the ratio of correctly matched questions to non-rejected questions, and the ratio of correctly matched questions to all test questions. We also track the total number of rejected questions and of incorrectly rejected questions (i.e. questions for which there really is a match in the cluster).

We train our model with a single hidden layer of size 100, over 25 epochs with a batch size of 1024. The input is composed of pre-trained $word2vec$ embeddings with 300 dimensions.

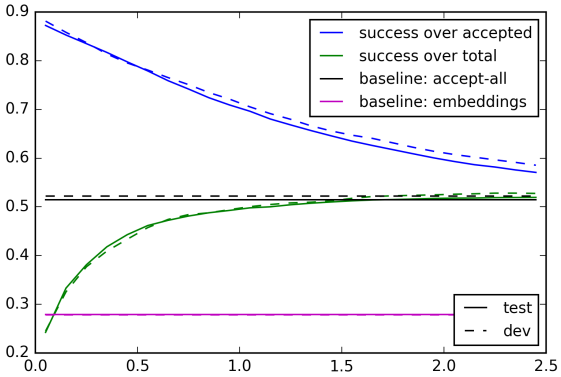

Fig. 2. Our accuracy for different values of β in comparison with our two baselines.

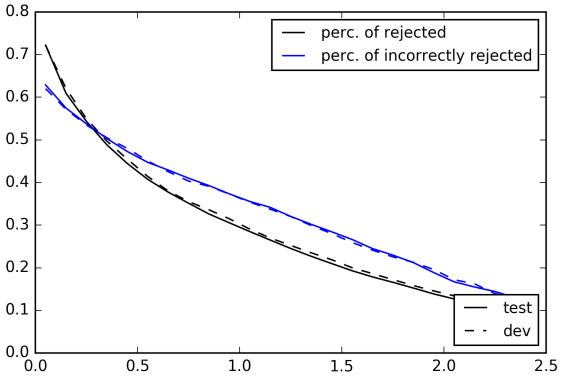

Fig. 3. % of rejected questions for different values of β and % of times we incorrectly rejected a question.

We compare our approach with two baselines: *embeddings* uses the cosine similarity of the average embeddings of the questions as a similarity metric and chooses the nearest question; *accept-all* is the case when β is infinite and no questions are rejected, in which case we always take the nearest question in the cluster.

As Figure 2 shows, the results are almost identical for the dev and test sets. We obtain an accuracy of 75% with $\beta = 0.75$, at the price of rejecting 35% of the questions, 60% of them correctly rejected: it would be impossible to find the right answer in that cluster. A solution without a threshold would force the model to make a choice even when the test instance is very different from the what the model has been trained on; the confidence scores of neural classification models become highly unpredictable for instances that are sufficiently different from the training data [22].

The dramatic improvement in accuracy over the *embeddings* baseline, which achieves an accuracy of only 27.8%, shows that our model's representation is very different from the sentence's lexical embeddings, and it does seem to constitute

- Who is the player who played for Miami Sol and went to school at North Carolina State?
- Which method resulting in a win against Ed Mahone?
- Tell me the host for midwest thomas assembly center.
- At which track was Frank Kimmel the Pole Winner of the Pennsylvania 200?

Fig. 4. Four questions that share a SQL template.

Fig. 5. Siamese LSTM representation for the Most frequent types of SQL queries

a sort of "semantic pattern space", as we hypothesize. At the same time, using our model without using a threshold - as the *accept-all* baseline does - yields an accuracy of only 51.5%. Overall, the experiments suggest that the best solution is to combine the ability of recurrent Siamese networks to accurately transform the questions to a semantic pattern space with a threshold to classify selectively.

Figure 4 shows four example questions that share a SQL template. Despite the lexical and semantic differences, our model correctly identifies the structural similarity and matches them; the embedding baseline does not.

A. Question Representation

As a secondary, informal evaluation of our approach, we visually inspect the quality of the embeddings produced by the LSTM. These embeddings are obtained by reading the state of the last hidden layer after processing a question with the trained LSTM. Consequently, each question is represented as a vector of dimension 100 (size of the hidden layer) and is projected to 2 dimensions using t-SNE.

Figure 5 shows the projected embeddings. The colors of the dots represent the true class (SQL template) of the question. We exclude groups with less than 500 members since they are difficult to evaluate visually.

VI. Conclusion

We show how to accurately and efficiently find the semantic structure of a sentence by comparing it with sentences with known structures, and evaluate our approach on questions aligned with SQL queries. Our results indicate that a combination of recurrent Siamese networks and nearest neighbor threshold validation yields high accuracy results.

References

[1] E. Agirre, M. Diab, D. Cer, and A. Gonzalez-Agirre, "Semeval-2012 task 6: A pilot on semantic textual similarity," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, 2012.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013.

[3] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014.

[4] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[5] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015.

[6] W. Guo and M. Diab, "Modeling sentences in the latent space," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-volume 1*, 2012.

[7] J. Zhao, T. Zhu, and M. Lan, "Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, pp. 271–277.

[8] H. He, K. Gimpel, and J. Lin, "Multi-perspective sentence similarity modeling with convolutional neural networks," in *EMNLP*, 2015.

[9] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *AAAI*, 2016.

[10] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract meaning representation for sembanking," in *The 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 2013.

[11] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[12] O. Biran, T. Blevins, and K. McKeown, "Mining paraphrasal typed templates from a plain text corpus," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[13] S. Sekine, "Automatic paraphrase discovery based on context and keywords between ne pairs," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[14] R. Barzilay and L. Lee, "Learning to paraphrase: an unsupervised approach using multiple-sequence alignment," in *NAACL-HLT*, 2003.

[15] G. Angeli, P. Liang, and D. Klein, "A simple domain-independent probabilistic approach to generation," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010.

[16] D. Duma and E. Klein, "Generating natural language from linked data: Unsupervised template extraction," in *Proceedings of the 10th International Conference on Computational Semantics*, 2013.

[17] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *SIGNLL*, 2016.

[18] K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang, "Generating sentences by editing prototypes," *arXiv preprint arXiv:1709.08878*, 2017.

[19] F. J. Och and H. Ney, "The alignment template approach to statistical machine translation," *Computational linguistics*, vol. 30, no. 4, pp. 417–449, 2004.

[20] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*, 1992, pp. 539–545.

[21] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994.

[22] W. Wang, A. Wang, A. Tamar, X. Chen, and P. Abbeel, "Safer classification by synthesis," *CoRR*, vol. abs/1711.08534, 2017.