Using online update of distributional semantics models for decision-making support for concepts extraction in the domain ontology learning task

A Anikin, A Katyshev, M Denisov, V Smirnov, D Litovkin

Volgograd State Technical University, Russia

anton@anikin.name, poas@vstu.ru

Abstract. Most of the information processed by computer systems is presented in the form of text corpuses. The number of such texts (as well as the corpus as a whole) only increases with time, and therefore the word processing tasks remain relevant to this day. Ontology allows to describe semantics using domain concepts and relations between them \[1, 2\]. In the ontology learning task, the ontology is dependent on quality of corpus which may not be readily available. There are different approaches to creating ontologies (including the use of different tools and frameworks). This paper discusses the use of word2vec (group of related models that are used to produce word embeddings) using online vocabulary update and extension of the original data corpus with additional training for the domain concepts extraction to automate the domain ontology creation.

1. Introduction

So far, word2vec \[3\] can’t increase the size of vocabulary after initial training. To handle unknown words, not in word2vec vocabulary, the model should be retrained with the updated documents. Definitely, the quality of the construction depends not only on the parameters of the model itself, but also on the very composition of the body of texts on which the model will be built \[4\]. Currently, there are a huge number of already constructed markup models, which are based on a huge corps of articles, books, etc. (for example, Google News word2vec model, Russian Distribution Thesaurus (RDT) Word2Vec model \[5\], etc). They contain ready-marked vectors of words and links between them.

But for a specific domain (for example, object-oriented programming), the data constructed models will be somewhat abstract and fuzzy. For the basic terms of our subject domain discussed in the models above, links with other words will be less clear and meaningful (for example, in Google News completely different sources from different domains are used, so some sense of the words this week will be somewhat blurred and distorted).

In this paper we consider the methods of constructing online dictionaries. It is based on the additional training of the already existing mark-up model with the corpus of texts of a specific domain that the user needs at the moment. We assume that when the already markup model of a huge corpus is completed with the corpus of texts of a specific domain (or with the imposition of its created model of this specific domain on the already prepared, markup model of a huge corpus), the links between the basic concepts of this specific domain will become more meaningful and adequate.

2. Background work

In this paper, we consider the additional training on the following ready-made markup models:

- **RDT-model**, built on the basis of a collection of books in Russian (150 GB text);
- **Wikipedia model** built on the basis of Russian Wikipedia articles dated October 1, 2018.

Also in this paper we consider the construction of a specific domain model: object-oriented programming, object-oriented design patterns. The basis of this subject domain is the following books:
Elementary design patterns [Elementarnye shablony proektirovaniya] (Jason McColm Smith);
Expressive JavaScript [Vyrazitelniy JavaScript] (Marain Haverbeke);
Corporate Application Templates [Shablony korporativnyh prilozheniy] (Martin Fowler);
Programming technologies [Tehnologii programmirovaniya] (Kamaev V. A.);
Object-oriented programming, analysis and design [Ob”ektno-orientoravannoe programmirovanie, analiz i dizain] (V. Mukhortov, V. Yu. Rylov);
Dive into design patterns [Pogruzhenie v patterny proektirovaniya] (Alexander Shvets);
Object-oriented design techniques. Design patterns [Priemy ob”ektno-orientoravannogo proektirovaniya. Patterny proektirovaniya] (E. Gamma, R. Helm, R. Johnson, J. Vlissides);
Design patterns on the NET platform [Patterny proektirovaniya na platforme NET] (Teplyakov S.);
Object-oriented thinking [Ob”ektno-orientoravannoe myshlenie] (Matt Weisfeld);
Object-oriented analysis and design with examples of applications in C++ [Ob”ektno-orientoravanniy analiz i dizain s primeryami prilozheniy na C++] (Buch G.);
Application of design patterns [Primenenie shablonov proektirovaniya] (J. Vlissides).

These books were used in a normalized format (all words are singular, masculine, lowercase: text without punctuation; each sentence starts on a new line).

To build a model of this specific domain based on the above books, the following approaches were used:

- “intersection” of the constructed model of a specific domain with a ready markup huge model (in the RDT-model) using the online dictionary;
- the extension of the original data corpus with additional training (Wikipedia model).

3. State of work

As mentioned above, to improve the quality of connections between terms and to improve the quality of highlighting related subject concepts, two approaches were used:

- online vocabulary update;
- extension of the original corpus data with additional training.

In this work, word2vec was used by Gensim. All ready-made markup word2vec models are stored as a KeyedVectors object - a module which implements word vectors and their similarity look-ups.

Table 1 shows the advantages and disadvantages of this module.

First, create a word2vec model of a specific domain based on the above books.
### Table 1. Using KeyedVectors vs. full model.

| Capability                  | KeyedVectors | Full Model | Note                                                                 |
|-----------------------------|--------------|------------|----------------------------------------------------------------------|
| Continue training vectors   | -            | +          | Need the full model to train or update vectors                       |
| Smaller objects             | +            | -          | KeyedVectors are smaller and need less RAM, because they don’t need to store the model state that enables training |
| Save/load from native word2vec format | +            | -          | Vectors exported by the Facebook and Google tools do not support further training, but you can still load them into KeyedVectors |
| Append new vectors          | +            | +          | Add new entity-vector entries to the mapping dynamically              |
| Concurrency                 | +            | +          | Thread-safe, allows concurrent vector queries                        |
| Shared RAM                  | +            | +          | Multiple processes can re-use the same data, keeping only a single copy in RAM |
| Fast load                   | +            | +          | Load data from disk instantaneously                                   |

```python
my_model = Word2Vec(size=500, sg=1, min_count=5, window=7, iter=5) files = os.listdir(data_path) for current, file in enumerate(files, start=1): update = current != 1 my_model.build_vocab(LineSentence(data_path + file), update) my_model.train( LineSentence(data_path + file), total_examples=my_model.corpus_count, epochs=my_model.epochs)
```

Results of the building of the model shows in table 2

### Table 2. Specific domain word2vec model.

| Compared words | Similarity       |
|----------------|------------------|
| klass + ob’ekt | 0.5569945351865275 |
| shablon + pattern | 0.7296937667116223 |
| izolirovat’ + inkapsulirovat’ | 0.6953800021098984 |
| pattern + stroitel’ | 0.5999575695010988 |
| pattern + fabrichniy | 0.47411129901295646 |

Consider both of these approaches in more detail and compare the results.
3.1. **Online vocabulary update**

According to the table 1, `KeyedVectors` does not support additional training of the markup model. To change the weights of the already marked vectors on the ready-made model, we can merge in an input-hidden weight matrix loaded from the original C word2vec-tool format, where it intersects with the current vocabulary. No words are added to the existing vocabulary, but intersecting words adopt the files weights, and non-intersecting words are left alone.

Below is the source code for online vocabulary update:

1. Load the input-hidden weight matrix from the original C word2vec-tool format and precompute L2-normalized vectors:
   ```python
   pretrained_model = KeyedVectors.load_word2vec_format(pretrained_vectors, binary=True, unicode_errors='ignore')
   pretrained_model.init_sims(replace=True)
   ```

2. Build vocabulary from a sentences of each book:
   ```python
   my_model = Word2Vec(size=500, sg=1, min_count=5, window=7, iter=5)
   for current, book in enumerate(books, start=1):
       is_update = current != 1
       my_model.build_vocab(LineSentence(books_path + book), update=is_update) total_examples = my_model.corpus_count
   ```

3. Update (expand) the created vocabulary with words from the ready-made markup model:
   ```python
   my_model.build_vocab(list(pretrained_model.vocab.keys()), update=True)
   ```

4. Merge in an input-hidden weight matrix loaded from the original C word2vec-tool format, where it intersects with the current vocabulary:
   ```python
   my_model.intersect_word2vec_format(pretrained_vectors, binary=True, lockf=1.0, unicode_errors='ignore')
   ```

5. Update the models neural weights from the corpus of books of specific domain:
   ```python
   for book in books: my_model.train(LineSentence(books_path + book), total_examples=total_examples, epochs=my_model.epochs)
   ```

Results of online vocabulary update shows in table 3

| Compared words            | Similarity     |
|---------------------------|----------------|
| `klass + ob"ekt`          | 0.42248948304496003 |
| `shablon + pattern`       | 0.43343555274649304 |
| `izolirovat' + inkapsulirovat'` | 0.39584950908535205 |
| `pattern + stroitel'`     | 0.2778668522863343  |
| `pattern + fabrichniy`    | 0.28110550019336966 |

3.2. **Extension of the original corpus data**

Extension of the original corpus data can be used when the original data corpus is available in the primary version (for example, there is the original Google news corpus, the original Wikipedia articles, the original set of books of artistic literature, etc.).

So, in this work we used the original set of articles of the Russian Wikipedia (version of October 1, 2018), taken from the Wikipedia dump.

All articles were converted to a single text file, on the basis of which the word2vec model was created. Then the constructed model was trained with the corpus of books of a specific domain.

Below is the source code for extension of the original Wikipedia corpus data:
(1) Loading the word2vec Wikipedia model:

```python
#wiki_model support online training and getting vectors for vocabulary words
pretrained_model = Word2Vec.load(wiki_model)
```

(2) Update vocabulary model from a corpus of texts of specific domain and update the models neural weights from that:

```python
for book in books:
    pretrained_model.build_vocab(LineSentence(books_path + book), update=True)
    pretrained_model.train(LineSentence(books_path + book),
                            total_examples=pretrained_model.corpus_count,
                            epochs=pretrained_model.epochs)
```

Results of extension of the original corpus data shows in table 4.

| Compared words | Similarity |
|----------------|------------|
| klass + ob"ekt | 0.7179621516632567 |
| shablon + pattern | 0.6129935413162676 |
| izolirovat' + inkapsirovat' | 0.5621600231928434 |
| pattern + stroitel' | 0.46872551920656125 |
| pattern + fabrichniy | 0.541209595734097 |

4. Comparison of the results

According to the results presented in the tables 2, 3 and 4 it can be concluded that the data obtained during the additional training of the RDT-model showed worse measures of similarity of key terms of the domain than the data obtained when creating a simple model of a specific domain.

This is primarily due to the original corpus of the books that were used to create the RDT-model. This model is based on a huge number of books of different types: encyclopedias, fiction, educational literature, articles, etc. Therefore, the results were less accurate than expected.

Especially worth noting the results obtained in the construction of an extended model of Russian Wikipedia. The similarity of terms such as klass and ob"ekt has increased significantly. However, the similarity of other pairs of terms (for example, shablon and pattern) decreased slightly.

We also analyzed the top most similar words from a variety of domain terms (such as klass, ob"ekt, shablon, pattern, inkapsulaciya, abstrakciya, etc). Several other terms turned out to be more similar to these words, however, they are synonymous with these word pairs, which were considered in the tables 2, 3 and 4.

The table 5 shows some of the most similar words to term inkapsirovat'.

| Similar word | Similarity |
|--------------|------------|
| skonfigurirovat' | 0.8240861892700195 |
| abstragirovat' | 0.816191554069519 |
| instancirovat' | 0.8075761795043945 |
| specificirovat' | 0.8035067915916443 |
| abstragirovat'sya | 0.8031325936317444 |

It is preferable to use Wikipedia models for analyzing a more specific domain and building its ontology on the basis of an already prepared markup model, since this model contains a more accurate description of articles.
5. Conclusions

Methods of additional training of ready-made markup models considered in this paper have both advantages and disadvantages (and in general they are interchangeable).

The main advantages of the online vocabulary update are:
- relatively simple using (we only need to build a basic dictionary of the specific domain we need, and make its intersection with the existing ready-made markup model);
- easy connection with any ready-made word2vec model (for online vocabulary update you can use any word2vec model with different parameters).

As the main disadvantage, we can highlight the absence of the necessary ready-made model. In the free access may not be the marked up model, on the basis of which we need to build a model of a more specific domain (for example, we need a ready-made word2vec model built on the corpus of all historical notes of the XVII century).

The main advantages of the extension of the original corpus data are:
- freedom in choosing the original corpus (we can independently select the necessary corpus);
- simpler opportunity for additional training (we do not operate with KeyedVectors, but with the full model, so we do not need to build basic vocabulary, but we can immediately proceed to train full model with our corpus of texts of the specific domain).

The main disadvantage is the long time for building the original large model because of the huge data corpus based on which this model is created (for example, it took a little less than a day to create a model of Russian Wikipedia articles: 10 hours to create a single text file with all articles of Russian Wikipedia and another 10 hours for direct creation of Russian Wikipedia word2vec model).

The results of the work can be used for the domain ontology learning task to automate the domain concepts extraction for the ontology [1].

6. Acknowledgements

This paper presents the results of research carried out under the RFBR grant 18-07-00032 "Intelligent support of decision making of knowledge management for learning and scientific research based on the collaborative creation and reuse of the domain information space and ontology knowledge representation model”.

References

[1] Anikin A, Litovkin D, Kultsova M and Sarkisova E 2016 Ontology-based collaborative development of domain information space for learning and scientific research Knowledge Engineering and Semantic Web: 7th International Conference, KESW 2016, Prague, Czech Republic, September 21-23, 2016, Proceedings 301–315
[2] Anikin A, Sychev O and Gurtovoy V 2019 Multi-level modeling of structural elements of natural language texts and its applications Biologically Inspired Cognitive Architectures 2018 1–8
[3] Mikolov T, Chen K, Corrado G and Dean J 2013 Efficient estimation of word representations in vector space vol abs/1301.3781 (Preprint 1301.3781)
[4] Venu S H, Mohan V, Urkalan K and TV G 2017 Unsupervised domain ontology learning from text 132–143
[5] Panchenko A, Ustalov D, Arefyev N, Paperno D, Konstantinova N, Loukachevitch N, Biemann C 2017 Human and Machine Judgements for Russian Semantic Relatedness Analysis of Images, Social Networks and Texts: 5th International Conference, AIIST 2016 221–235