

Data-driven measure of convergence for word embeddings

Martinez-Ortiz, Carlos Attema, Jisk Bos, Patrick
Kenter, Tom

July 7, 2017

Abstract

Word embeddings have been used in a wide range of applications. It is generally agreed that training a word embedding requires a significant amount of data, however it is never entirely clear how much data is enough for the word embedding to capture the semantic structure of the language it is being trained with. Specific tests, based on word analogies, have been used to decide when a word embedding has captured enough of the language structure to be considered meaningful. However this approach has several issues, amongst others the language specificity of such tests.

In this paper we propose a completely data-driven method for determining the level of convergence of a semantic model, which enables us to train semantic models with enough data for the model to capture the semantic structure of language and stop the training when the model does not benefit significantly from the additional data. The proposed method measures the amount of change induced by each additional batch of data used to train the model; the internal structure of the model will continue to change as long as the model is still learning. Once the internal structure of the model stops changing, then the model can be considered as *stable* and the training can stop.

The proposed method can be particularly useful when a limited amount of data is available or when the available data should be divided to train several models.

We compare our method with existing word analogy tests and show how our method still performs well in situations where word analogies are unable to perform as expected.

1 Introduction

Since their introduction [6], word embeddings have become increasingly popular and have been used for a wide range of applications [7, 3, 4, 1, 8]. Word embeddings are often trained using large data sets, such as the Google News corpus [https://code.google.com/archive/p/word2vec/], incorporating 100 billion words. The generally accepted approach when training word embeddings is

the more data, the better, and there does not appear to be any consensus on the minimum amount of data required to train a word embedding model which yields good results. A widely accepted method for evaluating word embeddings is by testing their performance in analogy tests: *man : woman :: king : queen*. Data sets for testing analogies are widely available (see [https://www.aclweb.org/aclwiki/index.php?title=Google_analogy_test_set_\(State_of_the_art\)](https://www.aclweb.org/aclwiki/index.php?title=Google_analogy_test_set_(State_of_the_art))). However there are several known issues with such tests [9]:

- **Language specific** - analogies are usually language specific – training a word embedding model on a language different from that of the analogies will obviously lead to failing the test;
- **Domain specific** - analogies usually fall in a particular knowledge category. For instance the following analogy: *Athens : Greece :: Baghdad : Iraq*, is specific to Geographical facts; a semantic model trained on a data set which does not contain such knowledge (which never mentions these locations) is bound to fail this test. At the same time, a word embedding may have already learnt a domain specific analogy, such as *electron : electricity :: photon : light*, but if this analogy is not reflected on the analogy test set, there is no way to know the semantic model already incorporates this knowledge;
- **Constrained by vocabulary** - because word embeddings typically have a set vocabulary; analogies which use out-of-vocabulary words, will always fail. Domain and language specific issues can be considered specific instances of constrained vocabulary issues.

These issues can be addressed, for instance, by translating the analogy test sets, discarding analogies which are irrelevant for the data being used, constructing analogies suitable for a domain, etc. However all of these approaches are labour intensive and must be hand crafted for each training data set. This work is motivated by the desire to find a data-driven method to validate word embeddings. The data-driven method introduced here does not rely on the vocabulary and thus does not require translation or creation of new test sets in order to function. This makes our approach effectively language independent.

The data-driven method for word embedding evaluation measures the amount of change in the semantic space at every stage of the training. Large variation in the semantic space indicates that the model has acquired a large amount of knowledge from a given batch of data. Small variation in the semantic space indicates that the given batch of data has contributed little new information to the model. Once the variation in the model falls below a given threshold, the model is assumed to have assimilated enough information from the data to be able to capture the semantic structure of the data set.

This method may also be particularly useful in cases where the available data needs to be divided to allow for diachronic data analysis. In such cases there is a clear trade-off between the volume of data used to train the model and the temporal resolution of the analysis: data slices spanning longer periods of time

contain more data and produce better models; on the other hand, data slices spanning shorter periods of time will allow a more detailed temporal resolution. The data-driven method allows us to find the right balance for such scenarios.

This paper is structured as follows: Section 2 describes the data-driven procedure for measuring convergence of word embeddings; Section 3 describes the setup used to evaluate word embedding convergence; Section 4 discusses the evaluation; concluding remarks are presented in Section 5.

2 Method

We will start by giving a brief overview of how semantic models work, more details can be found in the original papers [6]. A semantic model (M) works by creating a semantic space in which each word (w) in the known vocabulary (V) is represented by a vector. This semantic space has N dimensions. The number of dimensions is set at the beginning of the training process. The size of the vocabulary, is also determined from the start. Note that it is not unusual to restrict the vocabulary size to only include words which appear frequently (more than k times) in the corpus.

To find the semantic space vector — or *embedding* — of a word, we apply the following procedure:

Any given word w in the vocabulary can be represented in *vocabulary space* as a vector of size V using *one-hot* encoding. The linear transformation from vocabulary to semantic space is represented by an embedding matrix S of size $N \times V$. We can use matrix S to find the embedding of w , which we call e_w :

$$e_w = S \cdot w$$

where vector e_w is of size N . Words which have similar meaning will be located close by in this semantic space [TODO: ref paper found by Janneke here]. Word similarity is usually measured using cosine similarity [ref?]. The semantic model learns the structure of the semantic space by analysing sentences in the corpus and updating the S matrix.

We are interested in finding a matrix S which is a good representation of our data. As we add more data to our semantic model, there will be different versions of S changing over time: S_1, S_2, \dots, S_t . By measuring the amount of change between any two consecutive models, M_{i-1} and M_i , we can know how much the model needed to adapt to between $t = i - 1$ and $t = i$ to fit the new data presented to the model. As some point in time $t = t^*$, the difference between the two models, M_{t^*-1} and M_{t^*} , will be small enough that we can consider the two models to be equivalent.

We say that two models are equivalent when the local relations from one space are preserved in the other. This can be understood as that there exists a transformation to map between the two spaces.

In order to compare any two points along the training time, A and B , we use the two versions of their embedding matrices, S_A and S_B . We want to find a transformation matrix such that:

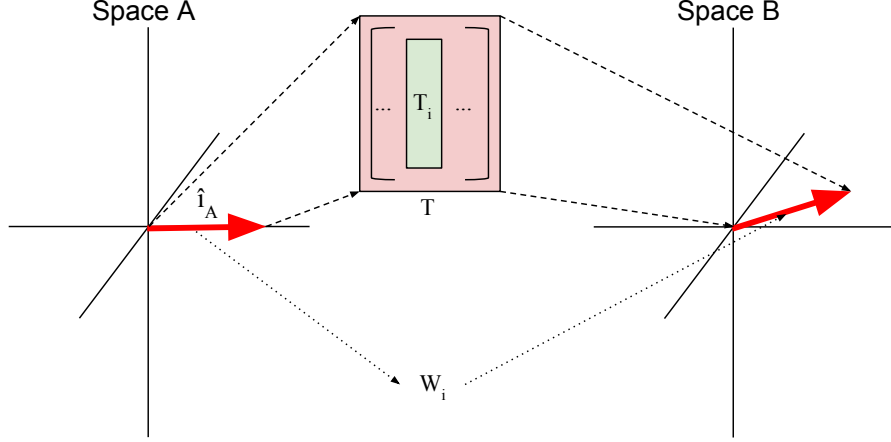


Figure 1: The i -th dimension transformation matrix $T (T_i)$, represents vector \hat{i}_A transformed from space defined by S_A to from space defined by S_B .

$$S_B \approx S_A \cdot T$$

The transformation matrix T can be found using the procedure described in Pseudocode 1:

```

foreach dimension  $i$  in the embedded space do
     $\hat{i}_A \leftarrow$  unit vector along the  $i$ -th dimension of the embedded space  $S_A$  ;
     $W_i \leftarrow \hat{i}_A$  projected back into vocabulary space a b ;
     $T_i \leftarrow W_i$  projected in embedded space  $S_B$  ;
     $i$ -th dimension of  $T \leftarrow T_i$  ;
end

```

Algorithm 1: Find T to map from S_A to S_B

^aTo convert \hat{i}_A back into vocabulary space, we take the product of the embedded vector and the inverse of embedding matrix.

^bIf the vocabularies are different we need to convert W_i to the other vocabulary T will be of size $N \times N$.

Alternatively, one can calculate T using the following equation for each column T_i of T :

$$T_i = S_B S_A^{-1} \hat{i}_A$$

where \hat{i}_A is the unit vector along dimension i in the semantic space defined by S_A .

Figure 1 depicts how the transformation matrix is built.

We will use the transformation T to measure how much the embedded space has changed between times A and B . To ensure that our measure is symmetric, we will calculate two transformation matrices, $T(A, B)$ to go from A to B and

$T(B, A)$ to go from B to A . We call the element-wise product of these two matrices the divergence matrix:

$$D(A, B) = T(A, B) * T(B, A)$$

It is important to mention that the divergence matrix will be equal the identity matrix if the space embeddings S_A and S_B are equivalent ¹. This follows from the fact that $T(X, Y)$ will be the identity when X and Y are equivalent. Therefore, given A and B are equivalent, $T(A, B)$ and $T(B, A)$ will both be the identity, and thus $D(A, B) = T(A, B) * T(B, A)$ will also be the identity. **NOTE: think / check if the element-wise multiplication makes sense, as pointed out by Patrick.**

We use $D(A, B)$ measure the change in the semantic space in two different ways:

- By taking the normalized trace of $D(A, B)$ - this will be equal to 1 if S_A and S_B are equivalent and yield a value between 0 and 1 if they are different. We call this *convergence score*.
- By looking at the values in the main diagonal of $D(A, B)$ - these values will be equal to 1 if S_A and S_B are equivalent and yield a value between 0 and 1 if they are different. This is similar to looking at the trace, but it provides additional information regarding the changes in the structure of the semantic space: are all vectors in the semantic space changing? Or are some of these vectors stable while others are changing significantly? We call this *diagonal convergence scores*, and they provide not only information regarding whether the semantic space is changing, but also how it is changing.

3 Experiments

In this section we describe the experimental process we have followed to validate our approach. For this experiment we used the Times news paper archives (**I need to ask Jose how can we reference the corpus**) between 1900 and 1980. We trained a semantic model, using gensim word2vec implementation, using CBOW [ref]. We set the dimensions on the semantic space to 300 ($N = 300$).

The vocabulary was limited to use a maximum of 1GB of memory; further, *nltk* was used to remove non-words and stop-words [ref nltk]. The resulting models had a maximum vocabulary size of 22,000 words. Notice that stop-word removal yielded a better final model with our data, but it does not affect the calculation of the divergence matrix.

We used batches of 10,000 sentences taken from randomly selected years in the corpus. After each batch is added to the model being trained, a snapshot of the embedding is taken. We continue to feed batches to the model until it has been trained with 10 million sentences.

We use $D(A, B)$ to compare snapshots. For comparison, we also use the Google "questions-words" analogy data set [ref]. It should be mentioned that

¹have we defined what do we mean by equivalent?

some of the analogy tests failed for every one of our trained models because some words in those tests were not present in our corpus (more specifically, they were not present with enough frequency to be part of the vocabulary in our models). For example, even after training with 10 million sentences, our model fails for the analogy: *London : England :: Montevideo : Uruguay*. This is however not surprising as *Montevideo* appears only a 87 times in the first million sentences (for reference, *London* appears in 21,000 sentences). Notice that the proposed measure calculates the difference between two embeddings, while the question words evaluates the individual accuracy of a given model.

4 Results

Figure 2 shows the convergence score (blue) as the number of sentences increases, and the Google accuracy (green). After 10^6 sentences have been used for training the model, there is a rapid increase in the convergence score. This indicates that the vectors defining the semantic space are becoming more stable. Training the model with additional sentences will produce little change in the model.

On the other hand, the Google accuracy score increases slowly but steadily. However, it requires a large number of sentences to increase the performance of the model. Depending on the application, there might be a trade off between the cost of using additional data and the benefit of doing so (notice that such a cost may not be economical but in terms of time, availability of data, or otherwise). In such cases it may be beneficial to stop once the cost/benefit balance crosses a certain point.

At around $2 * 10^4$ sentences, there is a local maxima on the convergence score, but its value drops down at around 10^5 sentences only to increase again at 10^6 sentences. Such behaviour is indeed confusing; here we use the diagonal convergence scores gain some insight.

Figure 3 shows the values on the diagonal of diagonal convergence scores as the number of sentences increases. Each line represents the convergence score for one particular dimension of the semantic space. What we see here is that around $2 * 10^4$ sentences there is wide variety of values for the convergence scores – some dimensions of the semantic space are relatively stable (high convergence scores) while others are undergoing significant change (low convergence scores). As more sentences are added to the model, the changes in semantic model become more widespread, until finally the model becomes globally stable. **TODO – Discuss: Why is Google Accuracy score so low ?**

4.1 Temporal corpus slicing

Semantic models have been used for diachronic analysis of vocabulary [5, 2] **Add more refs from [2]**. In such cases a corpus must be divided in temporal slices. Each of these slices will contain a subset of the corpus, with documents from a specific period (e.g. documents written between 1900 - 1905). This imposes a constraint in the volume of data available to train the model. In such

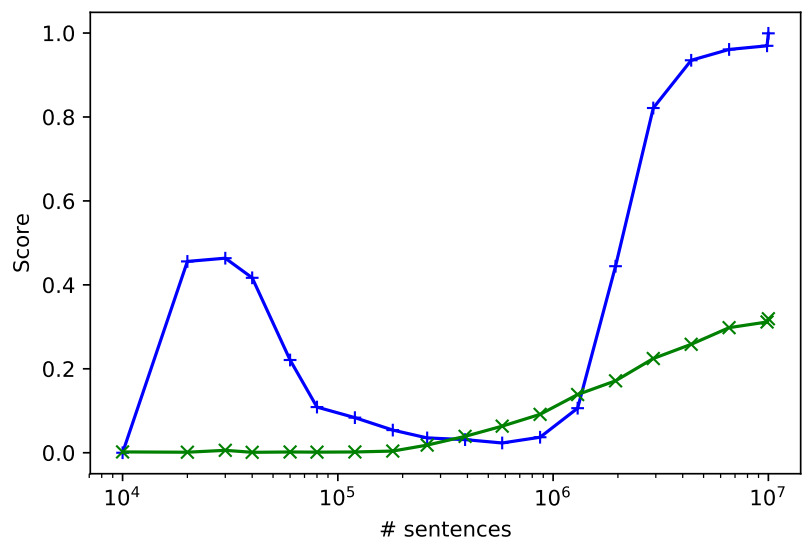


Figure 2: xxx

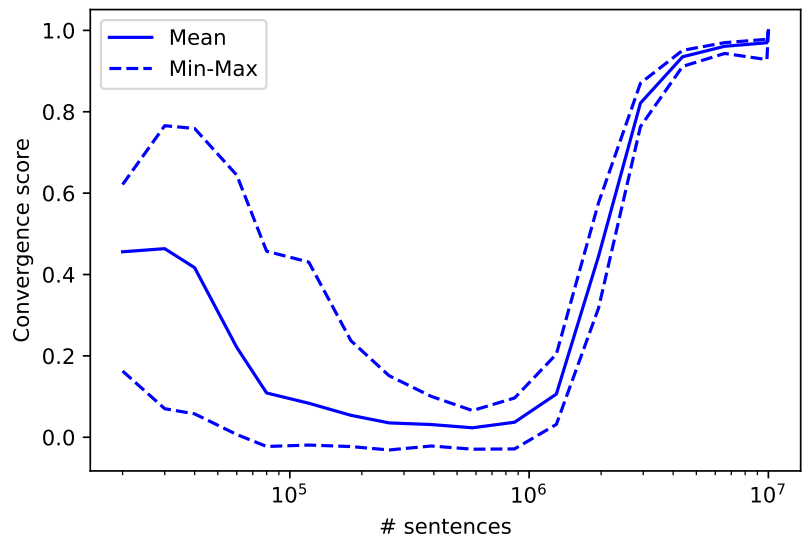


Figure 3: xxx

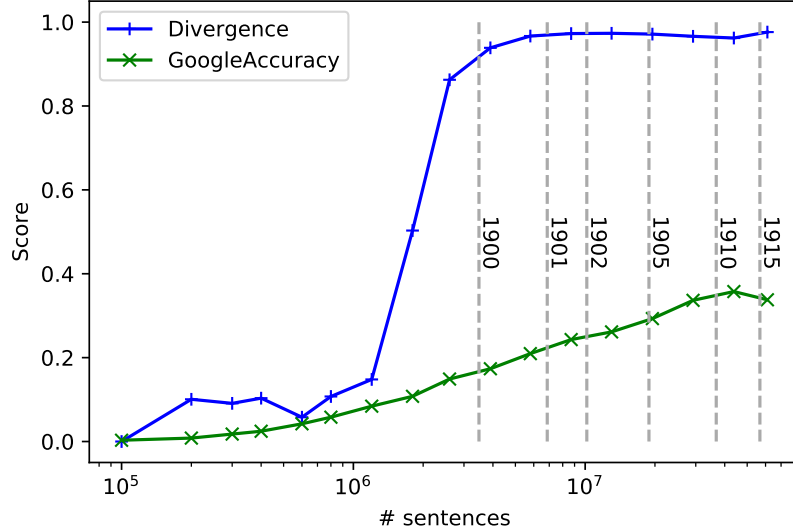


Figure 4: xxx

scenario, the trade off between accuracy and data availability becomes evident: slices covering shorter time spans (e.g. a single year) will be more useful for temporal analysis of the data (e.g. allowing to identify the year when a change in the language took place); at the same time, the model will have less data available for training, thus compromising accuracy.

In order to find the optimal number of years to be used for temporal slices on the Times corpus described above, we trained semantic models, adding data in chronological order from 1900 onwards, and taking snapshots every 1/2 million sentences. The following figure shows the convergence score for these models. The Google accuracy score is also included for reference.

As it can be seen, after the model has been trained with 5 million sentences, there is a noticeable increase in the convergence measure, indicating that the model has stabilized. This is a period equivalent to two years; from this we conclude that using two year temporal slices is suitable for this dataset.

5 Conclusion

In this paper we have presented a data-driven method for measuring convergence of semantic models. This approach determines when a semantic model has been trained with enough data to capture the semantic structure of the data. This is useful when there is a trade off between longer training and more accurate models.

The convergence scores can be visualized as a single value over time, as shown on Figure 2, or as a collection of values as shown on Figure 3. The first form of visualization is more straight forward, while the second provides additional insight into the structure of the semantic space as a whole.

This approach is entirely data-driven and does not depend on previous knowledge of the language being modelled or on a manually translated reference corpus. This will help remove previously existing limitations in the application of semantic models.

References

- [1] Silvio Cordeiro, Carlos Ramisch, Marco Idiart, and Aline Villavicencio. Predicting the compositionality of nominal compounds: Giving word embeddings a hard time. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.
- [2] Johannes Hellrich and Udo Hahn. Bad company - neighborhoods in neural embedding spaces considered harmful. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2785 – 2796, December 2016.
- [3] Yoon Kim, Yi-I. Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. Temporal analysis of language through neural language models. *CoRR*, abs/1405.3515, 2014.
- [4] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics.
- [5] Carlos Martinez-Ortiz, Tom Kenter, Melvin Wevers, Pim Huijnen, Jaap Verheul, and Joris Van Eijnatten. Design and implementation of shico : Visualising shifting concepts over time. In *HistoInformatics 2016*, volume CEUR Workshop Proceedings, 2016.
- [6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
- [7] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.
- [8] Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR*, abs/1702.03859, 2017.

- [9] Rob Speer. Cramming for the test set: We need better ways to evaluate analogies. Blog, June 2016.