Natural Language Engineering (2021), **27**, pp. 691–720
doi:10.1017/S1351324920000297

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Towards syntax-aware token embeddings

Diana Nicoleta Popa[1,2]*, Julien Perez[2], James Henderson[3] and Eric Gaussier[1]

[1]Laboratoire d'Informatique de Grenoble, Université Grenoble Alpes, 700 Avenue Centrale, 38401 Saint-Martin-d'Hères, France,[2]Naver Labs Europe, 6 Chemin de Maupertuis, 38240 Meylan, France, and [3]Idiap Research Institute, 19 Rue Marconi, 1920 Martigny, Switzerland
*Corresponding author. E-mail: diana.popa@imag.fr

(Received 16 January 2019; revised 11 December 2019; accepted 31 March 2020; first published online 8 July 2020)

Abstract

Distributional semantic word representations are at the basis of most modern NLP systems. Their usefulness has been proven across various tasks, particularly as inputs to deep learning models. Beyond that, much work investigated fine-tuning the generic word embeddings to leverage linguistic knowledge from large lexical resources. Some work investigated context-dependent word token embeddings motivated by word sense disambiguation, using sequential context and large lexical resources. More recently, acknowledging the need for an in-context representation of words, some work leveraged information derived from language modelling and large amounts of data to induce contextualised representations. In this paper, we investigate **S**yntax-**A**ware word **Tok**en **E**mbeddings (**SATokE**) as a way to explicitly encode specific information derived from the linguistic analysis of a sentence in vectors which are input to a deep learning model. We propose an efficient unsupervised learning algorithm based on tensor factorisation for computing these token embeddings given an arbitrary graph of linguistic structure. Applying this method to syntactic dependency structures, we investigate the usefulness of such token representations as part of deep learning models of text understanding. We encode a sentence either by learning embeddings for its tokens and the relations between them from scratch or by leveraging pre-trained relation embeddings to infer token representations. Given sufficient data, the former is slightly more accurate than the latter, yet both provide more informative token embeddings than standard word representations, even when the word representations have been learned on the same type of context from larger corpora (namely pre-trained dependency-based word embeddings). We use a large set of supervised tasks and two major deep learning families of models for sentence understanding to evaluate our proposal. We empirically demonstrate the superiority of the token representations compared to popular distributional representations of words for various sentence and sentence pair classification tasks.

Keywords: Token embeddings; Syntax-aware word representations; Tensor factorisation

1. Introduction

Word embeddings have been shown to be useful in a variety of tasks including part-of-speech (POS) tagging, named entity recognition (NER), machine translation, semantic role labelling and dependency parsing (Collobert *et al*. 2011; Zou *et al*. 2013; Bansal, Gimpel, and Livescu 2014; Kim 2014; Tang, Qu, and Mei 2015). They have been widely used in the context of deep supervised learning tasks in all the subfields of NLP. This success stems from the flexibility of vector space representations and the insights of the distributional semantic algorithms used to train them. But these algorithms are limited to learning about word types (entries in a dictionary), which has lead to a reliance on bag-of-words and sequence-of-words representations as the form of input to deep learning models. To convey more information about the linguistic analysis of the sentence

(e.g., word senses, syntactic-semantic structure) while keeping vectors as the input representation, a natural approach is to use embeddings of word tokens (contextualised words in a specific sentence).

The idea of *token* embeddings is to represent a word in its context, with the same word bearing different representations in different contexts. This contrasts with a generic word *type* embedding representation, which is the same in every context. For example, given two sentences containing the same words, but having different meanings:

$$S_1 : \textit{The cat chased the dog.}$$

$$S_2 : \textit{The dog chased the cat.}$$

we want to be able to distinguish between the two given their set of token embeddings. Ideally, we want these token embeddings to convey syntactic-semantic information, such as that in one case *cat* is a semantic agent and in the other it is a semantic patient. Various recurrent neural network (RNN) methods have been applied to this problem of computing vector representations from structures, but they are limited to sequence or tree structures, and more importantly, the structure itself is not necessarily embedded in the vectors.

Despite a growing interest in the topic of token embeddings computation, so far few pieces of work have tried to systematically evaluate the pertinence of such representations for natural language understanding tasks. It is very recently that some works have looked at the impact of using context-aware representations as input to deep learning models for text (Tu, Gimpel, and Livescu 2017; McCann *et al.* 2017; Salant and Berant 2018; Peters *et al.* 2018). However, these models do not make explicit use of the syntactic information, are mostly not unsupervised (except for Peters *et al.* 2018 and Tu *et al.* 2017) and generally leverage information coming from a pre-trained language model which in itself only considers the input as a sequence, without taking into account the structure of the sentence.

Nevertheless, early works such as that of Baroni and Lenci (2010) emphasise the important role played by syntactic structures (within structured distributional semantic models) in adjusting the distributional properties of words: having the word pairs connected through certain types of links should capture certain types of semantic relations between them. Additionally, a syntax-based approach should also help down-weigh the importance of relations between adjacent words that are not structurally connected through some dependency link (or for which such a link assumes a longer distance in the dependency path). Padó and Lapata (2007) also emphasise that syntactic relations are crucial for sentence processing, promoting the development and use of syntax-aware models as they capture more linguistic structure than linear contexts. Overall, the importance of considering syntax has been validated across much related work throughout time (Grefenstette 1994; Lin 1998; Levy and Goldberg 2014a; Weir *et al.* 2016).

While the current proposal fully supports the contextualisation of words based on their neighbouring context, we additionally provide an unsupervised way to inject explicit information from an arbitrary graph of a sentence that includes syntactic information. Thus, our token embeddings also capture information about the structure they are part of.

In this work, we propose to leverage a tensor factorisation method to adapt distributional representations of words to their sentential context. This method embeds information about an arbitrary graph of linguistic structure into the representation of each token in the sentence. The representations thus created are further denoted as **S**yntax-**A**ware word **Tok**en **E**mbeddings (**SATokE**), to stand for syntax-aware token embeddings. We use the linguistic structure to enhance word embeddings with the token's sequential and syntactic context and study the benefits of such a word representation on a large list of sentence and sentence pair classification tasks, using the two main families of deep learning approaches: RNNs and convolutional neural networks (CNNs).

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

The contributions of this paper are the following:

- an unsupervised model for token embeddings which reflect arbitrary graphs of linguistic context,
- a systematic evaluation of the token embeddings on a large series of tasks using two fundamental neural network architectures,
- an analysis of the contribution provided by **SATokE**.

The remainder of this paper is organised as follows: Section 2 outlines the main related work with a thorough discussion of existing methods for token embeddings computation and an in-depth parallel between them and the proposed method, Section 3 describes the process of computing the **SATokE** token embeddings, the architectures used for sentence representation, as well as the end task evaluation protocol. We also provide insights into how to encode any dataset using the proposed method, either by re-using pre-trained relations embeddings or by learning relations and token embeddings from scratch. Details about the datasets and tasks considered in our experiments, the pre-processing steps taken and implementation are provided in Section 4. Lastly, Section 5 is dedicated to an analysis of the results and Section 6 outlines the conclusions of the current work.

2. Related work

2.1 Leveraging syntactic dependency information within non-neural frameworks

While early work has leveraged syntactic dependency information in the construction of meaning representation (Grefenstette 1994; Lin 1998), in the following, we provide a detailed comparison to work that is most relevant and similar to the current proposal (Padó and Lapata 2007; Baroni and Lenci 2010; Weir *et al*. 2016).

Padó and Lapata (2007) choose to model meaning by considering the degree to which words occur in similar syntactic environments. They introduce a framework for semantic space models in which contexts are based on syntactic dependency information. The proposed framework also allows for considering words or combinations of words and syntactic entities as basis elements (dimensions) of the constructed space.

A first difference with respect to the current work lies in the choice of the structure they use to model information: Padó and Lapata (2007) use two-dimensional matrices for creating their dependency-based semantic spaces, while the current work proposes the use of three-dimensional tensors. As Baroni and Lenci (2010) point out, the mapping of three-way structured information (such as word-dependency-word) into a two-dimensional space can incur loss of information, making the approach of Padó and Lapata (2007) similar to an unstructured distributional semantic model.

Another difference with respect to the current work represents the choice of modelling the syntactic information along with the parts of speech of the head and modifier in the relation. This contrasts to our case in which we only consider the dependency relations themselves. Additionally, their model considers longer dependency paths, while our model focuses on direct links only. Nevertheless, both of these aspects could be easily incorporated in our proposal. Furthermore, the work of Padó and Lapata (2007) outlines the need for a flexible approach to incorporating linguistic knowledge. This is similar to the current proposal: our model allows for weighing differently the different syntactic relations and provides a flexible framework for incorporating additional knowledge.

Finally, unlike in the current proposal, the model of Padó and Lapata (2007) does not yield token embeddings: the co-occurrence counts are constructed over word types, not word tokens. Although an extension to word tokens would (at least in theory) be possible, it is

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

not straightforward how to practically implement this in their framework, especially given the significant increase in the parameter space.

Baroni and Lenci (2010) propose to model weighted tuple structures such as word-link-word co-occurrences as a large three-dimensional tensor. They experiment with several models, including one in which the links considered are based on syntactic dependency information. The tuples in their proposed model are weighted based on the local mutual information statistic. A first important distinction with respect to the current proposal refers to the dimensionality of their proposed structure: since in the case of Baroni and Lenci (2010) corpus-based co-occurrences are captured within one single tensor, the dimensionality is considerably superior to the one in the current proposal that considers multiple smaller tensors: their syntactic-based tensor has shape 30k by 800 by 30k, while on average, the current proposal deals with tensors of sizes 18 by 35 by 18.[a] Additionally, the tensors in the current proposal do not hold co-occurrence statistics across a large corpus but rather model binary information regarding the existence of local links between words of the same sentence.

Although the usage of a three-dimensional tensor makes the modelling of information conceptually similar to the current proposal, another important distinction is that the resulting word representations of Baroni and Lenci (2010) are still type embeddings rather than token embeddings (like in the current work): unlike our proposal, the model of Baroni and Lenci (2010) does not provide a different representation for each instance of a word depending on its context.

Moreover, throughout all their experiments, labelled tensor matricisation is performed to map the three-dimensional tensor into two-dimensional matrices in order to derive different semantic vector spaces representing either attributional or relational similarity. This contrasts to the current proposal in which we maintain the tensor-based representations of sentences. Only some preliminary results are provided using Tucker decomposition over a small part of the co-occurrence tensor, mainly due to computational costs. Nevertheless, like in the case of Padó and Lapata (2007), the resulting representations are still type-based.

Weir *et al.* (2016) also leverage information coming from syntactic dependencies: they address the issue of modelling semantic composition through the contextualisation of the words in a sentence by leveraging a common structure shared by them: the syntactic tree. In their proposal each word should have a fine-grained characterisation, derived from its particular context. However, their starting point is still represented by type-level co-occurrence statistics over syntactic dependency information which is different from our current approach. Later, these statistics are adjusted to account for the word's immediate context by down-weighing co-occurrences that are incompatible to the given context and up-weighing those that are. Although conceptually related, our model differs from theirs both in terms of modelling the use of syntactic information and in terms of the evaluation protocol.

2.2 From word embeddings to token embeddings

Much work has been successfully done for learning distributed representations of words (word embeddings) from vast amounts of data, some based on sequential contexts (Bengio *et al.* 2003; Mikolov *et al.* 2013b; Pennington, Socher, and Manning 2014), while others incorporating information from arbitrary contexts like the syntactic trees of sentences (Levy and Goldberg 2014a). One common problem however with such representations is that each word is only represented with one vector, making it challenging to capture polysemy and homonymy.

There have been several attempts to adapt these word embeddings to satisfy constraints extracted from different sources such as WordNet or BabelNet (Faruqui *et al.* 2015; Mrkšic *et al.* 2016; Vulic *et al.* 2017). The goal was to obtain more informed word representations, typically through a post-processing step starting from generic word embeddings. However, the resulting

[a]The average sentence length is 18, while 35 is the average number of syntactic relations we consider.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

representations were still generic in that they do not account for the specificity of each individual context the word appears in.

To obtain context-sensitive word embeddings, most previous work has typically treated this as a word sense disambiguation problem, with one vector per word sense (Neelakantan *et al*. 2014; Liu, Qiu, and Huang 2015; Chen, Liu, and Sun 2014). In these approaches, there is a fixed number of vectors to learn for each word which is based on the number of senses a word can have.

Recently, Melamud, Goldberger, and Dagan (2016) propose using an architecture based on CBOW (Continuous Bag-of-Words) (Mikolov *et al*. 2013a) where local context is considered by running two LSTMs (Long short-term memory networks) (Hochreiter and Schmidhuber 1997) on the right and left vicinity of a target word with the goal of having the context predict the target through a log-linear model. Their method does not use syntactic information and, although it improves on modelling the context of a target word, it does not provide one different representation for each word token. Dasigi *et al*. (2017) also propose a method to obtain context-aware token embeddings; however, they produce their embeddings by estimating a distribution over semantic concepts (synsets) extracted from WordNet. In contrast to their method, ours does not use any lexical resources and uses syntactic information from parse trees.

The method we propose provides a different word token vector for every individual context. One proposal related to the current work is that of Tu *et al*. (2017). They use a feed-forward neural network to produce token embeddings which they further evaluate as features for part-of-speech taggers and dependency parsers. However, they do not make use of the sentence parse tree to induce the syntactic information to the token embeddings, but rather use the local context (as defined by a window of words) to estimate the token representations. It is not clear how these representations behave in a sentence understanding scenario and how to explicitly add the syntactic information on top of their proposed model.

McCann *et al*. (2017) also provide context-aware word vectors (CoVe) by transferring a pre-trained deep LSTM encoder from a sequence-to-sequence model trained for machine translation (MT-LSTM) to a variety of NLP tasks. Although there is an intersection between some of the tasks they consider and the tasks considered in the current work, their use of a complex architecture in the form of a biattentive classification network makes any direct comparison to the results obtained using simple LSTM and CNN architectures unfair. Additionally, their input to the biattentive network is a concatenation of GloVe pre-trained word embeddings and the MT-LSTM context embeddings, both of which were trained on large corpora (CommonCrawl-840B for GloVe and three different English–German machine translation datasets for MT-LSTM). They show that the size of the corpora used to pre-train the MT-LSTM directly correlates with the results obtained on the end tasks to which these representations are transferred. This contrasts to our method as we don't directly combine the pre-trained word embeddings to our token representations in the end task evaluation, but merely condition on them in the earlier stages of computing the token embeddings. Additionally, our method to encode sentences does not rely on pre-training on large corpora.

Following on the same idea of transfer learning, some recent works leverage information coming from language models in semi-supervised settings. Peters *et al*. (2017) make use of the parameters learned in a pre-trained bidirectional language model to induce contextual information to token representations. The resulting tokens are used in a supervised sequence tagging model and tested on the tasks of named entity recognition and chunking.

Peters *et al*. (2018) extend the idea by learning a task-specific linear combination of the intermediate layers of a deeper bidirectional language model (ELMo) pre-trained on large corpora. Their representation of tokens is thus task-dependent in that, although the representations of the intermediate layers are task-independent, their linear combination is learned with respect to the task at hand. Additionally, the authors note that in most cases, even the parameters of the pre-trained bidirectional language model are fine-tuned on the training data, before being fixed during task training. In contrast to this, we make a clear distinction between computing the token representations (in an unsupervised manner with respect to the end tasks) and using them in any

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

subsequent evaluations. Similarly to McCann *et al.* (2017), their token representations obtained from the bidirectional language model are concatenated with pre-trained word representations before being fed into a task RNN, which enables them to directly use distributional information derived from large corpora. Moreover, contrary to our approach, all of the previous works (except for Tu *et al.* 2017) use character-level information in addition to or integrated within their models.

Salant and Berant (2018) show that adding contextualised representations derived from language modelling to a basic model for question-answering achieves state-of-the-art results. Moreover, they show that the model making use of contextualised token representations is able to surpass more sophisticated models that focus on the interaction between the question and the document. They also outline the need of complementing the information coming from the embedding of rare words with contextualised representations. Their findings validate further the importance of having contextually informed features as input to deep learning models, even when these models have simple architectures.

Recently, Devlin *et al.* (2019) have also provided a method (BERT - Bidirectional Encoder Representations from Transformers) to obtain token embeddings by leveraging an architecture based on multi-layer bidirectional transformers (Vaswani *et al.* 2017).

Within these recent models, it has been shown that the amount of data used for constructing representations correlates with an increase in performance, making training on large amounts of data a pre-requisite for the success of such approaches: CoVe (McCann *et al.* 2017) uses 350M words corpus for its best performing model, ELMo (Peters *et al.* 2018) uses a 1B words corpus, while BERT (Devlin *et al.* 2019) leverages a 3.3B words corpus and an architecture of 340M parameters. In contrast to these methods, the current proposal can provide competitive results even when leveraging only the corpus at hand to construct representations for its tokens.

2.3 Sentence representations

There have been many architectures proposed to model compositionality in variable-length texts, ranging from simple composition methods based on pooling operations over vector representations of individual words (Mitchell and Lapata 2010; Blacoe and Lapata 2012) to more complex compositional functions (Grefenstette and Sadrzadeh 2011).

A popular approach to encoding an arbitrarily long sentence as a single vector is to train a composition function which computes a sentence vector. Socher, Manning, and Ng (2010), Socher *et al.* (2012), İrsoy and Cardie (2014) use recursive neural networks, Kim (2014), Kalchbrenner, Grefenstette, and Blunsom (2014) use CNNs, Socher *et al.* (2011) use recursive auto-encoders over parse trees, and Zhao, Lu, and Poupart (2015) use a hierarchical sequence modelling approach to learn compositional representations over word representations.

Different variants of LSTMs and BiLSTMs (Bidirectional Long short-term memory) are also frequently used for sentence encoding. Pooling-based methods are commonly employed to obtain a sentence embedding from a set of word embeddings, either when implicitly used as an intermediary representation within neural networks-based approaches or when directly evaluated. Bowman *et al.* (2015) represent each sentence in a pair of sentences by a vector obtained by summing over the word embeddings in the sentence, while Liu *et al.* (2016) use an average pooling layer over word-level BiLSTMs to produce sentence vectors. Both use the obtained sentence embeddings as intermediary representations to solving further tasks.

Kiros *et al.* (2015) proposed an unsupervised approach for learning a sentence encoder by adapting the Skip-Gram model (Mikolov *et al.* 2013b) to sentences, called the SkipThought model. Using a large collection of novels, they train an encoder–decoder model such that a sentence is encoded to predict its surrounding context. They further evaluate the obtained representations on eight tasks by training linear models and show they provide robust performance across all tasks. More recently, Conneau *et al.* (2017) train universal sentence representations using natural language inference data and show improvement over previous sentence encoding models on a variety of transfer tasks.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

An alternative to modelling a sentence as a single vector is to use a number of vectors which grows with the length of the sentence (as in our proposed method). Neural network models of syntactic parsing can be seen as doing this, but also keep an explicit representation of the syntactic tree structure to determine which vectors to access for each decision (Henderson 2003; Socher *et al*. 2013a). In machine translation, Bahdanau, Cho, and Bengio (2014) represent the sequence of words of the source sentence as a sequence of vectors, which are then used in an attention-based neural network model to generate the target sentence for the translation. Accessing the different vectors with attention does not require an explicit representation of the source sequence, but the method for encoding the source sentence as vectors does require that it be a sequence. It is not at all clear how to generalise such a sequence-encoding method to arbitrary graphs, as in our proposed method.

2.4 Spectral methods

One approach to learning representations which does generalise to arbitrary graphs is tensor factorisation. Such methods represent each node of the graph with a separate vector, so the number of vectors grows with the size of the graph. Tensor factorisation models have been widely used for learning representations of multi-relational databases (Nickel, Tresp, and Kriegel 2011; Trouillon *et al*. 2016). Furthermore, such methods have been also already proposed for inducing word representations (Levy and Goldberg 2014b).

3. SATokE – Syntax-based token embeddings via tensor factorisation

Following the success of tensor factorisation models at learning latent representations of entities and relations from a graph, we propose a model that makes use of tensor factorisation to produce token embeddings for sentences and their syntactic structures (SATokE). These token embeddings will be compared empirically to context-independent word embeddings (Pennington *et al*. 2014) in Section 5. We also provide a comparison to word embeddings trained on syntactic information (Levy and Goldberg 2014a) and to an alternative token embeddings method proposed in the literature ELMo (Peters *et al*. 2018).

3.1 General considerations

We propose to compute token embeddings in an unsupervised manner by choosing to model each sentence as a graph: the nodes in each graph represent the individual tokens in the sentence and graph edges stand for the relations between these tokens. In the current work, we consider dependency relations, as given by a parse tree of the sentence, as well as an adjacency relation that helps model the local context. The details on how the syntactic information is obtained are presented in Section 4.

To learn these token embeddings, we make use of tensor factorisation, through a model that can be seen at the intersection between the RESCAL model (Nickel *et al*. 2011) and the TransE model (Bordes *et al*. 2013), both of which have been proposed for learning embeddings in large relational databases. The structure used to model relations between entities in the current work follows that proposed in RESCAL, with entities being represented by vectors and relations by matrices. Similarly to TransE, we choose to optimise a ranking loss function, unlike RESCAL that proposes a squared loss optimisation in its original version.

However, in the current approach, entities are partitioned into sentences and no relations can exist between entities belonging to different sentences, which makes the learning more challenging. Unlike other multi-relational approaches where a threshold can be set on the number of times entities appear in relations in order to ease the learning (Bordes *et al*. 2013), this is not a feasible option in our case. The maximum number of times an entity takes part in a relation is limited

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

to a very low value which is dependent on the parse tree. The goal of learning in our model is to abstract away from the individual sentences and learn the underlying regularity of parse trees. Thus, unlike in a typical relational learning scenario, the proposed model learns from many small graphs rather than from one single big one.

The factorisation model works by optimising a reconstruction of the graphs' tensors, so that given the token embeddings one can reconstruct each labelled edge in the graph. To predict the different relations, it learns one real-valued matrix per relation label. It is these relation matrices which capture the regularities which generalise across sentences. The information about the individual sentence is captured in the token embedding vectors. Because each token, in conjunction with the relation matrices, needs to be able to reconstruct its relations with other tokens, the token vectors encode information about the graph context in which they appear. In this way, each token representation can be interpreted as a compressed view of the sentence's entire graph from the point of view of that token. The final sentence representation is a set of all individual token representations which, by construction, are contextually aware and syntactically aware.

Additionally to the tensor decomposition, our model incorporates another part of the loss that constrains the nature of the entity representations that we wish to obtain. In order to account for the semantic aspect, to enable information sharing across sentences and to make use of the distributional information inferred from a large corpus, we constrain each token embedding to be similar to a pre-trained representation of the word type it denotes.

One alternative is to learn the word type embeddings from scratch as explained in Section 3.2.2. However, one advantage of using pre-trained word type embeddings is the possibility to learn token embeddings for any dataset regardless of its size. Word type embeddings represent abstractions over the occurrences and uses of words in the datasets they are learned from (Westera and Boleda 2019). Thus, constraining the token representations to be close to pre-trained word type embedding values enables leveraging distributional information learned from large amounts of data, even when the tokens themselves are part of a smaller size corpus. Another consequence of such an approach represents the reduction in computational cost that would otherwise be attributed to learning these semantic representations from scratch.

3.2 Unsupervised learning of syntax-based token embeddings

In the following, we present the method for the computation of SATokE token embeddings. This method can be applied to any dataset of sentences with the sole prerequisites of having access to parsing information and to pre-trained general purpose word type embeddings.

There are two possibilities for token embedding computation. In the *1-step* setting, given a new dataset of sentences and their parses, we learn a representation for each word token in the sentences as well as for each relation holding between these tokens from scratch. We expect the token embeddings obtained through such a learning scenario to perform best as they are directly learned for the given dataset along with the relations present in the dataset. An alternative is the *2-steps* setting. This approach reduces the computation time as it leverages relation embeddings previously learned on a different corpus. However, it may also affect the quality of the induced token representations as the relations are kept fixed to values previously learned. Given a new dataset of sentences, their parses and previously trained relations embeddings, one can learn token embeddings for the dataset at hand through the same process by fixing the relation embeddings to the pre-trained ones and optimising only for the token embeddings. Further details on how we implement both settings are given in Section 4.3.

For completeness, we will further present the method corresponding to the *1-step* setting, but we will discuss results corresponding to the *2-steps* setting as well in Section 5. We hypothesise (and later show empirically) that the token embeddings learned in the *1-step* setting obtain better results than those learned in the *2-steps* setting, yet token embeddings learned in both settings outperform general purpose word type embeddings on the sentence understanding tasks considered.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

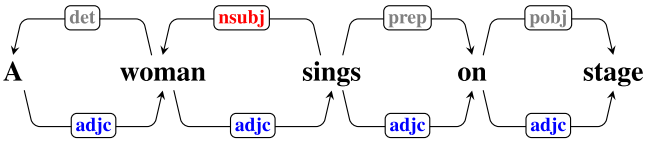

Figure 1. Example of relations that hold for one sentence: relations from the syntactic dependency parse (in grey and red) + adjacency relations (in blue): det (determiner), nsubj (nominal subject), prep (prepositional modifier), pobj (object of preposition), adjc (adjacency). The nsubj relation is marked in red to emphasise our focus within further analysis.

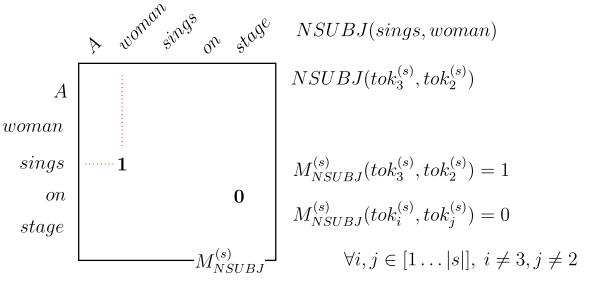

Figure 2. Example of a matrix for the dependency relation NSUBJ for sentence *s*.

3.2.1 Building the sentence tensors

The starting point for computing the SATokE token embeddings is represented by parsing the sentences in the given corpus and identifying the relations that hold between tokens. An example of a sentence is provided in Figure 1. For the given example, the syntactic dependency relations, as given by the parser, are marked in grey and red: **det**, **nsubj**, **prep**, **pobj**. An additional relation considered is the adjacency between tokens that is marked in blue in Figure 1.

The next step is building one three-dimensional tensor $T^{(s)}$ for each sentence graph $G^{(s)}$: nodes in the graph represent tokens in the sentence and edges between them stand for the relations that hold between tokens. The tensor $T^{(s)}$ models the interactions between the tokens in the sentence, with each matrix in the tensor representing a different interaction type: for a total of r possible binary relations between tokens, 1 matrix in $T^{(s)}$ corresponds to adjacency information, while the remaining $r-1$ matrices correspond to $r-1$ syntactic dependency relations. Each of the matrices is of dimensions $|s| \times |s|$ and is asymmetric.

Let $tok_i^{(s)}$ represent the token that appears at position i in the sentence s and let $M^{(s)}_{REL}$ denote the matrix in $T^{(s)}$ that holds entries related to the *REL* relation in sentence s. Then an entry t_{ijk} in the final tensor $T^{(s)}$ takes value 1 if the relation k holds between the token $tok_i^{(s)}$ and the token $tok_j^{(s)}$ such that the token $tok_i^{(s)}$ is the head in the dependency relation and $tok_j^{(s)}$ is the dependant. The matrix corresponding to the adjacency information is populated similarly with value 1 whenever the token at position i precedes the token at position j in the given sequence. All the remaining entries are set to value 0.

$$t_{ijk} = 1 \Leftrightarrow M_k^{(s)}(i,j) = 1$$

$$t_{ijk} = 0 \Leftrightarrow M_k^{(s)}(i,j) = 0$$

For the given example in Figure 1, the *NSUBJ* relation holds between $tok_3^{(s)}$ and $tok_2^{(s)}$, with $tok_3^{(s)}$ being the head of the relation. Thus, the matrix corresponding to the *NSUBJ* relation in sentence s will have value 1 at position $M^{(s)}_{SUBJ}(tok_3^{(s)}, tok_2^{(s)})$ and 0 everywhere else, as shown in Figure 2. Similarly, all the other matrices corresponding to syntactic dependency relations are populated

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

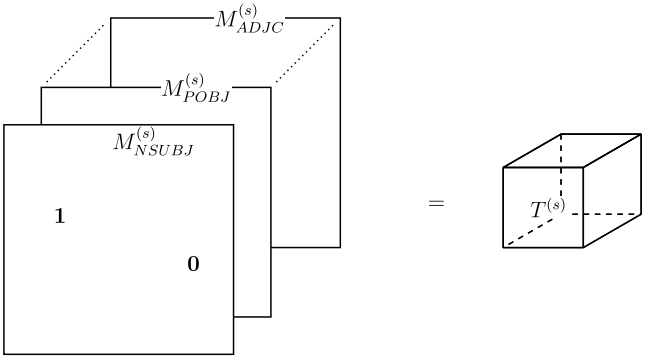

Figure 3. Matrices form the tensor $T^{(s)}$ for sentence s.

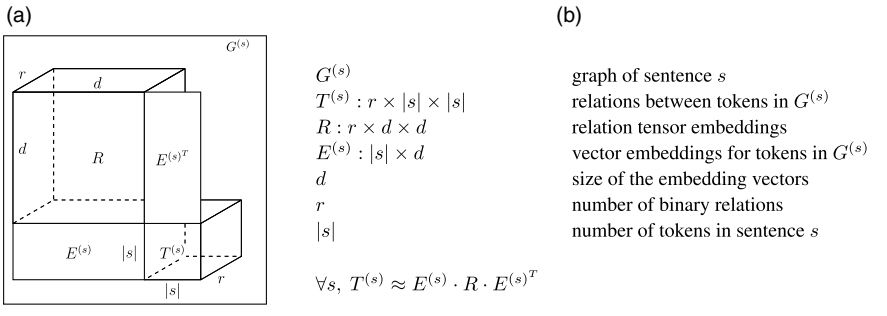

Figure 4. (a) Sentence graph decomposition for sentence s; (b) Legend.

based on the information from the sentence dependency parse tree. The adjacency matrix for sentence s holds value 1 at position $M^{(s)}_{ADJC}(tok^{(s)}_i, tok^{(s)}_{i+1})$ and 0 everywhere else, $\forall\, i \in [1 \ldots |s|\,]$.

For sentence s, the matrices corresponding to syntactic dependency relations along with the adjacency matrix form the tensor $T^{(s)}$ are shown in Figure 3. Such a tensor is formed for each of the sentences in a given dataset.

3.2.2 Decomposing the sentence tensors

In order to obtain embeddings for all the tokens in a sentence s as well as for all the relations holding between these tokens, we factorise each $T^{(s)}$ into:

- $E^{(s)}$, a matrix holding all token embeddings for sentence s, with one token embedding per row
- R, a tensor for all the relations embeddings (shared between all sentences), having one matrix per relation embedding.

Through the factorisation, the goal is to find token and relation embeddings from which we can recompute the various relations between tokens in each sentence:

$$T^{(s)} \approx E^{(s)} \cdot R \cdot E^{(s)^T}$$

The decomposition is shown in Figure 4.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

As objective function, we optimise a ranking loss within the tensor ($T^{(s)}_{loss}$) that aims at scoring the reconstruction of positive triples $t^{(s)}_{ijk} = (e^{(s)}_i, R_k, e^{(s)}_j)$ higher than that of negative ones. We thus relax the exact reconstruction constraint to a ranking one. Apart from the speed-up gain, adopting such an approach can be justified conceptually: since the aim is to reconstruct positive links in the graph, it is sufficient to ensure that the reconstruction of positives scores higher than that of negatives, without imposing constraints on the actual values these reconstructed entries should take.[b] Adopting a ranking approach is also common in related work on embedding multi-relational data (Bordes *et al*. 2013).

We use an additional regularisation term ($R^{(s)}_{loss}$) to minimise for each token, the gap between the token embedding representation and the word type representation of the word it denotes. Conceptually, this is similar to the vector space preservation term in Mrkšic *et al*. (2016) that controls how much the token embeddings can deviate from their corresponding word representations. The overall goal is to create embeddings that are close, through $R^{(s)}_{loss}$, to the original word embeddings (known to capture semantics) and at the same time are syntactically informed, through $T^{(s)}_{loss}$, so as to capture fine-grained semantic differences according to the role a given word plays in a sentence. Optimising only $T^{(s)}_{loss}$ would be insufficient as it would lack the notion of semantics provided through $R^{(s)}_{loss}$. Indeed, by imposing that the token embeddings be close to the word embeddings, one also forces the semantic representation of the word embeddings to be shared across all sentences.

The optimisation problem is formulated as

$$\min \sum_{s \in S} \alpha \left(T^{(s)}_{loss} \right) + (1 - \alpha) \left(R^{(s)}_{loss} \right) \tag{1}$$

where:

$$T^{(s)}_{loss} = \sum_{\substack{t^{(s)}_{ijk} \in G^{(s)}, \\ t^{(s)}_{i'j'k'} \in \neg(t^{(s)}_{ijk})}} \max \left(0, \gamma + \langle e^{(s)}_{i'}, R_{k'}, e^{(s)}_{j'} \rangle - \langle e^{(s)}_i, R_k, e^{(s)}_j \rangle \right)$$

and

$$R^{(s)}_{loss} = \sum_{e^{(s)}_i \in G^{(s)}} - \log \sigma \left(e^{(s)}_i \cdot w^{(s)}_i \right)$$

with S denoting the set of all sentences in the corpus, $G^{(s)}$ the graph of sentence s holding all tokens and all relations present in the sentence, $e^{(s)}_i$ the embedding of the token at position i in sentence s, R_k the matrix embedding for the relation k, $w^{(s)}_i$ the pre-trained word embedding corresponding to the token $e^{(s)}_i$, γ the margin hyperparameter and $\neg(t^{(s)}_{ijk})$ the set of negative triples associated with t_{ijk} as explained in Section 3.2.3.

d denotes the dimensionality of the embeddings, while r denotes the number of relations. Thus, the bilinear product $\langle a, b, c \rangle = a^{1 \times d} \cdot b^{d \times d} \cdot c^{d \times 1}$. Considering all tokens in one sentence s, that results in approximating the relations tensor for sentence s, $T^{(s)}$ by $E^{(s)(|s| \times d)} \cdot R^{(d \times r \times d)} \cdot E^{(s)^T (d \times |s|)}$.

[b] Initial experiments in which we adopted an exact reconstruction loss resulted in both an increased training time and a lower performance of the obtained embeddings.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

The regularisation term $R^{(s)}_{loss}$ can be seen as a particular case of cross entropy:

$$R^{(s)}_{loss} = -y \cdot \log \hat{y} - (1-y) \cdot \log (1-\hat{y})$$

with $y = 1$ and $\hat{y} = \sigma(e^{(s)}_i \cdot w^{(s)}_i)$, where σ denotes the sigmoid function.[c]

As mentioned in Section 3.1, instead of leveraging pre-trained word type embeddings, the current model could also be used to compute such representations given a large amount of data. To achieve that, one has to consider redesigning the loss to optimise: currently, the token embeddings are constructed to stay close to a semantic representation derived from large corpora (the type embeddings). In the lack of such a pre-computed representation, one option would be to compute it as an additional term in the current loss.

3.2.3 Methodology details

As the sentence tensors are sparse, during decomposition we consider all positive examples and only a subset of the negative ones. To do so, we explore multiple settings by varying the ratio of negative sampling for one positive triple. A negatively sampled example in the tensor is obtained by altering one element of the triple while fixing the remaining two: this element can be either one of the entities or the relation holding between them. As mentioned above, given a triple $t^{(s)}_{ijk} = (e^{(s)}_i, R_k, e^{(s)}_j)$, we denote by $\neg(t^{(s)}_{ijk})$ the set of negative examples associated with it. We consider here that $\neg(t^{(s)}_{ijk})$ is formed of the following elements:

$$\neg(t^{(s)}_{ijk}) = \left\{ \left(e^{(s)}_{i'}, R_k, e^{(s)}_j\right), \left(e^{(s)}_i, R_{k'}, e^{(s)}_j\right), \left(e^{(s)}_i, R_k, e^{(s)}_{j'}\right) \right\}_{\forall i' \neq i, j' \neq j, k' \neq k}$$

We optimise Equation (1) using mini-batch stochastic gradient descent. We construct our batches b_m such that if $t^{(s)}_{ijk} \in b_m$ then $\neg(t^{(s)}_{ijk}) \subset b_m$. Sampling positive points and their negative counterparts together is important to ensure the consistency of the information at the basis of which a step is taken when optimising each mini-batch.

At the end of the decomposition process, we obtain for each sentence s in a corpus S, a set of SATokE token embeddings $e^{(s)}_i$ that constitute the token-based representation of the sentence.

3.3 Sentence representations

The set of SATokE token embeddings $e^{(s)}_i$ which result from the factorisation presented in Section 3.2 is further fed as input to a neural network architecture. In this work, we have explored two main approaches proposed in the literature. This first one is based on an LSTM encoding of the sentence and the second one uses a CNN encoder. Similarly to recent literature, we also explore positional encodings and self-attention on top of word embeddings as ways to contextualise them using fully differentiable methods.

Given a sentence representation as a sequence of n tokens $(t_1, ..., t_n)$, let $(h_1, ..., h_n)$ be the hidden representations computed by the network in the case of **LSTM**. We further consider the sentence encoding to be its last hidden state representation:

$$S_{LSTM} = LSTM(t_1, ..., t_n) = h_n$$

For the **CNN** architecture, we follow the approach of Kim (2014). For a sentence $S = (t_1, ..., t_n)$, let $t_{i:i+j}$ be the concatenation of tokens $t_i, t_{i+1}, ..., t_{i+j}$. Let w be a filter applied to a window of h

[c] An alternative was to use L2 between the token embeddings $e^{(s)}_i$ and their corresponding word type embeddings $w^{(s)}_i$, but preliminary experiments resulted in decreased performance.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

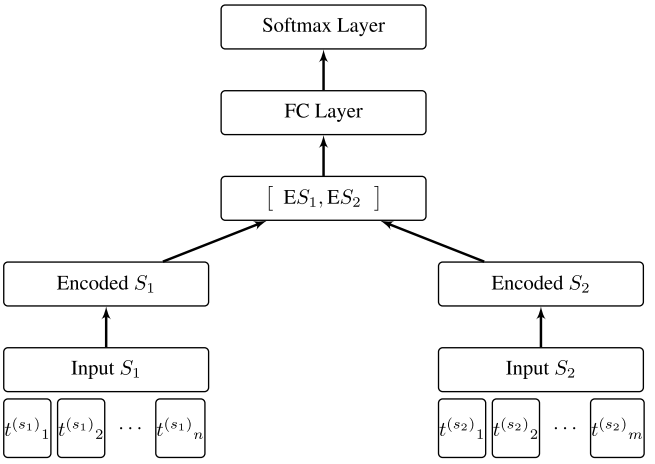

Figure 5. General architecture for sentence pair classification.

tokens to produce a feature c. Let b be a bias term and f a non-linear function. Then

$$c_i = f(w \cdot t_{i:i+h-1} + b)$$

A feature map is created by applying the filter to each possible window of tokens in the sentence $S : c = [c_1, c_2, ..., c_{n-h+1}]$ This is followed by a max-pooling operation $\hat{c} = \max(c)$. For m filters with different window sizes, we obtain m features: $z = [\hat{c}_1, ..., \hat{c}_m]$. Finally, these features are passed to a fully connected layer that outputs the probability distribution over labels: $y = w \cdot z + b$.

We report details on the parameters used for both LSTM and CNN in Section 4.3.

One way to inject into word representations information about their order in a sequence is through **positional encodings** (Gehring *et al.* 2017). Following the approach of Vaswani *et al.* (2017), we use fixed sinusoidal positional encodings of the same dimension as the word embeddings and sum them together. Given an input sequence of word embeddings $S = (w_1, w_2, ..., w_n)$, with $w_j \in \mathbb{R}^d$ and the positional information for each of the words $P = (p_1, p_2, ..., p_n)$, with $p_j \in \mathbb{R}^d$, our new sequence representation will be $S' = (w_1 + p_1, w_2 + p_2, ..., w_n + p_n)$. For the positional encodings, we use the sine and cosine functions:

$$p(pos, 2i) = \sin\left(pos/10000^{2i/d}\right)$$

$$p(pos, 2i+1) = \cos\left(pos/10000^{2i/d}\right)$$

with *pos* representing the position, i the dimension and d the embedding size.

Self-attention has also been recently used in a variety of tasks. It has been shown to be a useful mechanism to connect information from different positions of a sequence in order to better represent it. Similarly to Vaswani *et al.* (2017), we use a dot-product attention:

$$Att(Q, K, V) = softmax(QK^T)V$$

where Q, K and V are transformations of the input embeddings $(w_1, w_2, ..., w_n)$ through matrices W_Q, W_K and W_V. Then, the obtained vectors v_i computed through the attention mechanism constitute the input to the LSTM and CNN architectures.

3.4 End tasks evaluation

The general architecture for sentence pair classification can be seen in Figure 5. The architecture for single sentence classification follows the same structure as the one in Figure 5, but for one

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

single sentence as input. Sentences are encoded using the methods described in Section 3.3. For the sentence pair classification, the representations of the two sentences are concatenated to be provided as input to a fully connected layer before the softmax decision layer.

Through this architecture, we explore the behaviour of the produced SaTokE token embeddings on a large and representative set of tasks: textual entailment prediction, paraphrase detection, sentence polarity detection, subjectivity detection and question-type categorisation. We comparatively explore the behaviour of the ELMo token embeddings (Peters *et al*. 2018) and that of word type embeddings (Pennington *et al*. 2014; Levy and Goldberg 2014a) on the same set of tasks.

4. Experimental set-up

In the following, we detail the datasets used by presenting some of their characteristics and a discussion over the 'appropriateness' of considering them for evaluation in Section 4.1. Further, the pre-processing steps taken and the parsing information used are presented in Section 4.2. Finally, we give the implementation details for the SATokE token embeddings learning and the end task evaluations as well as some details about the word type embeddings we compare to, in Section 4.3.

4.1 Datasets

We test the quality of the computed embeddings when provided as input features in several tasks leveraging eight datasets, often considered in the literature (Conneau *et al*. 2017; Conneau and Kiela 2018). We examine their behaviour on sentence classification datasets: binary classification for sentiment analysis on movie reviews (MR, SST-2) and product reviews (CR), for subjectivity/objectivity classification (SUBJ) and multi-class classification on question types (TREC).

The **MR** dataset (Pang and Lee 2005) is a dataset of movie reviews with one sentence per review and with the goal to classify the reviews into two classes depending on the sentiment they express. The dataset is balanced with 50% the reviews bearing positive sentiment and 50% **SST-2** (Stanford Sentiment Treebank-2) (Socher *et al*. 2013b) is an extension of the MR dataset, with the train, development and test splits provided and having class distribution of 48% (negative)/52% (positive). The **CR** dataset (Hu and Liu 2004) contains customer reviews of various products (cameras, MP3s, etc.) in which the goal is to determine the polarity of the reviews: 63% of the reviews in the dataset are positive and 36% negative.

SUBJ (Pang and Lee 2004) is a balanced dataset containing subjective and objective sentences where the goal is to classify each sentence in one of the two classes. For the subjective sentences, Pang and Lee (2004) collected 5000 movie review snippets (e.g., 'bold, imaginative, and impossible to resist'), while the objective data were collected from plot summaries. **TREC** (Li and Roth 2002) is a question dataset that involves classifying a question into one of six question types, denoting whether it refers to a location, a person, a description, a numeric value, an abbreviation or an entity. The data are almost evenly distributed among the six classes: only the abbreviation class is underrepresented (0.01% of the data) and each of the rest of the classes is covered by approximately 20% of the cases. While for SST-2 and TREC, we respect the provided train/test splits, for MR, CR and SUBJ, the reported results are based on threefold cross-validation. Some sentence examples along with their corresponding classes are presented in Table 1. Some data statistics for the sentence understanding tasks considered are presented in Table 3.

We also explore sentence pair classification tasks on three datasets: binary classification for paraphrase detection (MSR) and three-class classification for textual entailment detection (SNLI, SICK).

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 1. Examples of sentences for the sentence understanding tasks considered

Corpus	Example	Class
MR	'one from the heart.'	positive
CR	'the little bag it comes with is cheap and useless.'	negative
SUBJ	'bruce is a down on his luck tv news reporter.'	objective
TREC	'what is the most popular sport in japan?'	entity
SST-2	'a vivid cinematic portrait.'	positive

MSR (Microsoft Research Paraphrase Corpus) (Dolan and Brockett 2005) is a collection of 5800 sentence pairs extracted from news sources on the web and human-annotated for paraphrase/semantic equivalence, 66% of which are paraphrase pairs. The train (4076)/test (1725) split is provided. **SICK** (Sentences Involving Compositional Knowledge) (Bentivogli *et al*. 2016) is a corpus of 10k sentence pairs, drawn from image and video descriptions and split into: entailment, neutral and contradiction, with a train/dev/test split provided. **SNLI** (Stanford Natural Language Inference) (Bowman *et al*. 2015) represents a benchmark for the evaluation of systems on the task of textual entailment. It consists of 570k human-written English sentence pairs labelled as entailment, contradiction or neutral and divided into train, development and test sets. In the current work, we use the development (20k) set as a train set and keep the provided test (20k) set for testing (further referred to as **SNLI-20k**). Table 2 presents examples of sentence pairs from all three datasets along with their corresponding labels. Some data statistics for the sentence pair classification tasks considered are presented in Table 4.

From Tables 3 and 4, it can be seen that the datasets with the highest type/token ratio, and hence those having the richest vocabulary are (in order) TREC, SUBJ, SST-2, MR, CR and MSR, with SNLI-20k and SICK having the least diverse vocabulary.

4.1.1 Data appropriateness

Although these datasets have been often used in literature (Conneau *et al*. 2017; Conneau and Kiela 2018) providing a consensus for natural language understanding evaluation, it is important to note some characteristics that might make them more or less appropriate for evaluation in the current set-up.

Table 5 presents some data statistics of the considered datasets: average sentence length and percentage of sentences in the dataset for which a subject and/or an object relationship has not been detected by the parser. Based on the sentence length alone, these datasets can be split into two categories: datasets containing short sentences with an average sentence length of around 10 words (TREC, SICK and SNLI-20k) and datasets having longer sentences with an average sentence length around 20 words (MR, CR, SUBJ, SST-2 and MSR). In general, one would expect the shorter sentences to be 'easier' to parse and thus *'appropriate'* for evaluation in a setting strongly based on parsed data. More precisely, by minimising the propagation of errors from the parsing stage into the token embeddings creation, one could obtain more accurate SATokE representations.

Another important observation from Table 5 is that many datasets contain a high percentage of sentences in which dependency relations such as SUBJ or OBJ are not detected or do not exist. After further analysis into such cases, we have concluded this is partly due (15%) to a parser deficiency (for the cases when the SUBJ relationship is not detected or is mistakenly labelled) and partly due to the actual data (for the cases where the subject is implicit). For example, as MR and SST-2 are both datasets of movie reviews, it is often the case that sentences present within these

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 2. Examples of sentence pairs used for the evaluation of paraphrase detection (MSR) and textual entailment recognition (SICK, SNLI-20k). *S*1, *S*2 denote the two sentences involved (or not) in a paraphrase relation. *H* stands for the entailed hypothesis and *T* stands for the entailing text in a textual entailment relation

Examples – Corpus	Class
MSR	
S1: 'His mother contacted the federal public defender's office in Sacramento, which has agreed to handle his surrender, she says.'; S2: 'His family approached the federal defender's office in Sacramento about arranging his surrender.'	paraphrase
S1: 'July 1st is the sixth anniversary of hong kong's return to chinese rule.'; S2: 'The rally overshadowed ceremonies marking the sixth anniversary of hong kong's return to china on 1 july 1997.'	not paraphrase
SICK	
T: 'An old woman is shaking hands with a man.'; H: 'Two persons are shaking hands.'	entailment
T: 'A baby is licking a dog.'; H: 'A dog is licking a baby.'	neutral
T: 'A squirrel is lying down.'; H: 'A squirrel is running around in circles.'	contradiction
SNLI	
T: 'A man looking over a bicycle's rear wheel in the maintenance garage with various tools visible in the background.' ; H: 'A person is in a garage.'	entailment
T: 'A couple walk hand in hand down a street.' ; H: 'The couple is married.'	neutral
T: 'A person dressed in a dress with flowers and a stuffed bee attached to it, is pushing a baby stroller down the street.' ; H: 'A lady sitting on a bench in the park.'	contradiction

datasets denote short descriptions of movies with implicit subjects. A similar situation is present in some examples from SNLI-20k due to the way the dataset was constructed to include instances from an image caption corpus. However, in the case of the SNLI-20k dataset, most of the cases in which the SUBJ relation is not detected are due to parser deficiency as the SUBJ relation is not detected or is mislabelled. One example from the MR dataset is presented in Table 1: the sentence considered lacks both a main verb and a SUBJ relation. Further examples from the MR, SST-2 and SNLI-20k datasets are present in Tables 6 and 7.

Being part of a SUBJ and/or an OBJ relation provides important information about the role of a word in a sentence which can further influence the representations of the rest of the graph. Therefore, having many sentences in a dataset for which such information is lacking or fails to be detected can affect the performance of the representations obtained on the basis of the parse tree. Out of all the datasets considered, the one that has the lowest percentage of such sentences

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 3. Data statistics for the sentence understanding datasets considered: number of classes (# classes), number of sentences (# instances), total number of unique word types in the corpus (# types), total number of word tokens in the corpus (# tokens), percentage of tokens in the corpus that are replaced by the UNKNOWN_POS tag (% oov), ratio between the total number of word types and the total number of word tokens as a measure of vocabulary richness: the larger the value, the more the vocabulary is diverse (ratio div)

Corpus	# classes	# instances	# types	# tokens	% oov	ratio div
MR	2	10.6k	18521	237939	15.49%	0.07
CR	2	3.7k	5388	77638	16.49%	0.06
SUBJ	2	10.0k	21098	251715	15.79%	0.08
TREC	6	5.9k	9673	60154	3.04%	0.16
SST-2	2	9.5k	18007	234879	7.92%	0.07

Table 4. Data statistics for the sentence pair classification datasets considered: number of classes (# classes), number of sentences (# instances), total number of unique word types in the corpus (# types), total number of word tokens in the corpus (# tokens), percentage of tokens in the corpus that are replaced by the UNKNOWN_POS tag (% oov), ratio between the total number of word types and the total number of word tokens as a measure of vocabulary richness: the larger the value, the more the vocabulary is diverse (ratio div)

Corpus	# classes	# instances	# types	# tokens	% oov	ratio div
MSR	2	5.8k	18804	258114	2.14%	0.07
SICK	3	10k	2407	191188	0.06%	0.01
SNLI-20k	3	40k	9593	464821	0.56%	0.02

Table 5. Data characteristics. Underlined are the *'appropriate'* datasets.

	MR	CR	SUBJ	TREC	SST-2	MSR	SICK	SNLI-20k
average sentence length	22.34	20.61	25.17	10.1	19.89	22.24	9.71	11.84
%sent without SUBJ rel	21.00%	8.00%	10.00%	12.00%	21.00%	2.00%	6.00%	24.00%
%sent without OBJ rel	11.00%	15.00%	4.00%	25.00%	14.00%	3.00%	7.00%	10.00%

without a SUBJ or OBJ relation is MSR. We retain this dataset as being *'appropriate'* for evaluation in the current set-up despite its high average sentence length.

The above discussion shows that one can split the datasets used for evaluation according to whether they are adapted to syntactic analysis, in which case we refer to them as *'appropriate'* datasets, or not, in which case we refer to them as *'inappropriate'* datasets. The *'appropriateness'* of a dataset is therefore judged based on whether the sentences contained within it are well formed and enable leveraging syntactic information or not, based on the criteria in Table 5. Having such a distinction should help assess the quality of the proposed representations while minimising the chances of propagating parsing errors. We consider the sentence length to be a better predictor of *'appropriateness'* as it is parser-independent and it represents a measure of complexity the parser needs to deal with. Contrary to this, the percentages of SUBJ or OBJ relations detected are parser-dependent and susceptible to parser errors: detecting a SUBJ or OBJ relation does not necessarily ensure correctness of the labelling. Therefore, we further consider datasets TREC, MSR, SICK and

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 6. Examples of sentences for which a SUBJ dependency relation is not detected due to the fact that the subject is implicit: the parts marked by [] are added artificially to emphasise the missing information

Corpus	Example
MR	'[It is] not too fancy, not too filling, not too fluffy, but definitely tasty and sweet.'
MR	'[There is a] great character interaction.'
MR	'[This is] a solidly entertaining little film.'
SST-2	'[It represents] uneasy mishmash of styles and genres.'
SST-2	'[It is] tailored to entertain!'
SST-2	'[It was] remarkably accessible and affecting.'
SNLI-20k	'A man at a restaurant.'
SNLI-20k	'A women's soccer game.'
SNLI-20k	'Dog in pool.'

Table 7. Examples of sentences for which the SUBJ dependency relation is not labelled as such due to the fact that sentences are not grammatically correct

Corpus	Example
SNLI-20k	'Two men on bicycles competing in a race.'
SNLI-20k	'A young boy in a field of flowers carrying a ball.'
SNLI-20k	'Family members standing outside a home.'

SNLI-20k to fall under the *'appropriate'* datasets category and the remaining MR, CR, SUBJ and SST-2 to constitute *'inappropriate'* datasets.

4.2 Pre-processing and linguistic information

We compute token embeddings for all sentences in the datasets we use. For that, we parse the sentences in all considered datasets and further encode them using the method described in Section 3.2. The dependency information used in the current work is obtained with the spaCy toolkit,[d] leveraging the transition-based dependency parser of Honnibal and Johnson (2015) and using the CLEAR label scheme for the syntactic dependencies.

When computing the token embeddings, we set a threshold on the frequency of words th_W to take into account (here set to 2). All words that appear in the corpus with a frequency lower than the threshold or that do not have a corresponding pre-trained GloVe embedding are set to an 'UNKNOWN_POS' tag, where POS stands for the part of speech associated with the word. For these words, we also learn representations along with the token representations. We also impose a threshold on the frequency of syntactic dependencies th_D to take into account (here set to 1,000). Data from infrequent relations are not discarded, but rather merged under a single 'UNKNOWN_RELATION' tag and learned along with the rest of the graph. The 'UNKNOWN_RELATION' tag is thus represented by an 'UNKNOWN_RELATION' matrix in each sentence tensor.

[d] https://spacy.io/.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

The percentage of words per dataset that are considered unknown and are thus replaced by the 'UNKNOWN_POS' tag are shown in Tables 3 and 4. While for some datasets such as TREC, MSR, SICK and SNLI-20k this percentage is rather low, the MR, CR, SUBJ and SST-2 contain a higher number of words that we tag as UNKNOWN due to their lack of a pre-trained embedding or their low frequency in the dataset. This consequently means that for such datasets methods that consider character-level information such as ELMo are expected indeed to perform better than our proposed method.

4.3 Implementation details

SATokE token embeddings unsupervised learning. For each dataset, we run multiple threads of token embeddings computation with varying negative sampling initial learning rate values in the range $[10^{-2}, 10^{-5}]$ and the factors (1x, 5x, 10x). We set the margin $\gamma = 1$, the ratio between the two losses $\alpha = 0.5$ and mini-batches of size 300. We re-sample negative triples for each positive triple at every epoch. For assessing the embeddings, we use the mean reciprocal rank (MRR), which is a standard metric for the evaluation of the quality of triple ranking. For the optimisation, we use Adam (Kingma and Ba 2014) and we do early stopping based on the MRR score on the validation set. All token embeddings are randomly initialised.[e]

To construct the validation set, we randomly sample 20% of the positive triples in the training set $t_{ijk}^{(s)} = (e_i^{(s)}, R_{k''}, e_j^{(s)}) \in T_+$ such that at least one of the two following conditions holds

$$\exists R_{k''}, e_{j''}^{(s)} \in G^{(s)} : (e_i^{(s)}, R_{k''}, e_{j''}^{(s)}) \in T_+$$

$$\exists e_{i''}^{(s)}, R_{k''} \in G^{(s)} : (e_{i''}^{(s)}, R_{k''}, e_j^{(s)}) \in T_+$$

where $G^{(s)}$ denotes the graph of the sentence composed of all the entities and relations present in the sentence and T_+ is the set of all positive triples in the dataset. This prevents us from separating the graph of a sentence into two disjoint parts for training and validation. It also allows us to learn embeddings for all entities as it enables information from the entities in the training set to flow to the entities in the validation set via the relations.

***1-step* versus *2-steps* learning set-ups**. For each dataset, we try two different learning strategies: either learn token embeddings and relation embeddings from scratch (*1-step* approach) or pre-train the relation embeddings on a corpus and then fix them to infer token embeddings for all the datasets considered (*2-steps* approach).

As mentioned in Section 3.2, one can leverage pre-trained relations embeddings in order to compute token embeddings for any new dataset. In order to assess the quality of token embeddings induced using such pre-trained relations embeddings, we employ a *2-steps* process: *(step-1):* compute relation embeddings on a corpus (different than the target corpus for which token embeddings are wanted). This is done through the decomposition presented in Figure 4, at the end of which the matrices of relations embeddings are saved. *(Step-2):* compute token embeddings on the target corpus through the same decomposition, yet this time fixing the relation embeddings to the values obtained in *(step-1)* and optimising only for the token embeddings in each desired dataset. In the current work, we first use our model to train relations embeddings on the Penn TreeBank corpus (Marcus, Marcinkiewicz, and Santorini 1993). For training the relation embeddings, we set the margin parameter $\gamma = 1$, we vary the initial learning rate value in the range $[10^{-3}, 10^{-4}]$, we set the ratio between the two losses $\alpha = 0.5$, mini-batches of size 300 and the negative sampling factors (1x, 5x, 10x). We then compute token embeddings for each dataset using the same hyperparameters as the ones used for learning the relations they are inferred from.

[e] Preliminary experiments with initialisation to GloVe did not result in improvements on the end tasks.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

End task supervised learning. Throughout, some of the runs we fix the input embeddings to be either the pre-trained word type embeddings (Pennington *et al.* 2014; Levy and Goldberg 2014a) or the SATokE or ELMo (Peters *et al.* 2018) token embeddings and train the network for each individual task. All experiments corresponding to such settings are denoted by the *(fixed)* keyword. This approach has the benefits of preventing an adaptation of the embeddings to the end task and disallowing the representations of frequent words in the training data to distance themselves in the embedding space from representations of related but rare words. This also enables us to perform a stricter assessment of the difference between the various type embeddings and token embeddings when provided as input features to different tasks. For the embeddings that lead to the best obtained results, we run an additional set of experiments in which we allow the input embeddings to change, further denoted as *(fine-tuned)*.

The pre-trained word type embeddings of Pennington *et al.* (2014) considered in the current work are 300-dimensional and were learned from the Common Crawl corpus.[f] These are used for constraining the semantics of the SATokE token embeddings through $R^{(s)}_{loss}$ as discussed in Section 3.2. They are also used for evaluation on each dataset in Section 5. The pre-trained word type embeddings of Levy and Goldberg (2014a) are also 300-dimensional and are learned using the Skip-Gram model (Mikolov *et al.* 2013a, b), from English Wikipedia, considering dependency-based contexts.[g]

We provide results of experiments using two variants of the ELMo token embeddings (Peters *et al.* 2018): the 1024-dimensional embeddings that obtained the best results in Peters *et al.* (2018) as well as the 256-dimensional embeddings obtained from the pre-trained ELMo models[h] (for fair comparison to our 300-dimensional SATokE). We use the pre-trained ELMo models as they represent the optimised versions shown to provide the best results (Peters *et al.* 2018). The ELMo models considered were trained on the 1B Word Benchmark consisting of approximately 800M tokens of news crawl data from WMT 2011.

To tune all the parameters for each end task, we use a development set when available or randomly select 20% of the training set when such a set is not provided. We vary the parameters using values often considered in the literature: the initial learning rate takes values in the range $[10^{-3}, 10^{-5}]$, the number of filters for the CNN architecture in {128, 256}, the dropout keep probability takes values in [0.6, 0.8] and the number of hidden units for the LSTM architecture in {128, 150, 512}. We consider filters of sizes 3, 4, 5 for the CNN architecture. For the optimisation, we use Adam and report the parameters that lead to the best results for each task in Table 8. We stop the optimisation when the loss increases on the development set.

5. Results and discussion

We present the results obtained using each of the architectures in Tables 9 and 10. The first five columns correspond to results for the sentence classification datasets while, the last three correspond to sentence pair classification tasks. The division into *'appropriate'* (last four columns) and *'inappropriate'* (first four columns) datasets is done according to the analysis in Section 4.1.1. We compare results using multiple input features: word type embeddings pre-trained on large corpora on linear contexts GloVe (Pennington *et al.* 2014), word type embeddings pre-trained on dependency contexts Dep-based WE (Levy and Goldberg 2014a) and our proposed token embeddings, SATokE. While more complex models can be used for each individual task, the current work is exploratory in that it aims to assess the importance of using token embeddings compared to standard pre-trained word embeddings along with the added value of using syntax on

[f] https://nlp.stanford.edu/projects/glove/.
[g] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/
[h] https://allennlp.org/elmo.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 8. Best run parameters for each task: α denotes the learning rate, *dp* the dropout keep probability, *nbh* the number of hidden states, *nbf* the number of filters and *fsz* the filter sizes

Corpus	LSTM runs			CNN runs			
	α	dp	nbh	α	dp	nbf	fsz
'Inappropriate' datasets							
MR	10^{-5}	0.7	512	10^{-4}	0.7	256	3,4,5
CR	10^{-5}	0.7	150	10^{-4}	0.8	256	3,4,5
SUBJ	10^{-5}	0.7	512	10^{-4}	0.7	256	3,4,5
SST	10^{-5}	0.7	512	10^{-4}	0.7	128	3,4,5
'Appropriate' datasets							
TREC	10^{-5}	0.7	512	10^{-4}	0.8	256	3,4,5
MSR	10^{-5}	0.7	150	10^{-4}	0.7	128	3,4,5
SICK	10^{-5}	0.7	512	10^{-4}	0.7	128	3,4,5
SNLI-20k	10^{-5}	0.7	150	10^{-4}	0.7	256	3,4,5

the different tasks. Additionally, we compare our SATokE token embeddings to the state-of-the-art token embeddings method ELMo, which is more complex and was trained on larger amount of data. We provide results using the original version of ELMo in which embeddings are fine-tuned during the end task learning (further referred to as ELMo-F) as well as a version in which we fix the embeddings for the end task learning.

5.1 Word types versus word tokens

We observe a consistent improvement in the performance of the SATokE token embeddings over pre-trained word type embeddings across all datasets when using a CNN architecture. The preferred architecture for word type embeddings such as GloVe is however the LSTM. Overall, SATokE surpasses GloVe embeddings in all but two cases of the *'inappropriate'* data (datasets MR and SUBJ when we allow the fine-tuning of GloVe embeddings). SATokE obtains the best results surpassing all considered word type embeddings on the *'appropriate'* data, with the top scores obtained for TREC, MSR and SNLI-20k when using a CNN architecture and for SICK using the LSTM.

Most improvements obtained by fine-tuning GloVe embeddings can be observed when using an LSTM architecture (datasets MR, SUBJ, SST, TREC and SNLI-20k) and only some hold for a CNN architecture as well (datasets MR and TREC). However, it is only in the case of the MR dataset that fine-tuning GloVe embeddings enables surpassing the score obtained using fixed SATokE embeddings. Overall, no such tendency can be observed in the case of fine-tuning SATokE embeddings.

As outlined in Table 9, the impact of using positional encodings and self-attention on top of word type embeddings is limited. Fixed positional encodings on top of word embeddings do not always help: we observe improvements only on two datasets (TREC and MSR) when using GloVe embeddings and a marginal improvement on the TREC dataset when using dependency-based word embeddings. Nevertheless, these improvements are not sufficient to exceed the accuracy provided by our proposed token representations. Self-attention improves the scores mostly on the sentence pair classification tasks when using a CNN architecture on top of GloVe embeddings:

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 9. Sentence classification and sentence pair classification results in terms of accuracy (%). Comparison of the results obtained using the proposed syntactically aware token embeddings (SATokE 300d) to results obtained using standard pre-trained word type embeddings: GloVe (Pennington *et al*. 2014) and Dep-based WE (Levy and Goldberg 2014a), including results using positional encodings and self-attention. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**

Architecture/Type Embeddings	*'Inappropriate'* datasets				*'Appropriate'* datasets			
	MR	CR	SUBJ	SST-2	TREC	MSR	SICK	SNLI-20k
LSTM								
Standard pre-trained word embeddings								
GloVe 300d (fixed)	75.86	76.79	92.06	80.42	80.6	68.63	61.96	57.13
GloVe 300d (fine-tuned)	**77.24**	73.23	**92.78**	**81.8**	82.4	68.28	60.33	**57.51**
GloVe 300d + pos encod	73.27	74.42	90.8	79.6	81.2	**69.5**	61.35	55.74
GloVe 300d + self-attention	75.44	**77.37**	90.92	79.49	70.4	68.17	58.76	50.62
Dep-based WE 300d	67.81	69.27	87.07	72.2	68.4	67.88	59.88	50.55
Dep-based WE 300d + pos encod	50.3	63.78	69.96	50.63	68.0	66.66	60.13	33.53
Dep-based WE 300d + self-attention	54.88	68.9	86.95	73.17	68.0	67.47	60.59	47.79
Current proposal token embeddings								
SATokE 300d (fixed)	74.39	73.0	91.19	79.6	**84.4**	69.27	**63.59**	56.76
SATokE 300d (fine-tuned)	72.13	66.91	89.99	76.58	54.2	69.15	59.68	51.51
CNN								
Standard pre-trained word embeddings								
GloVe 300d (fixed)	74.92	76.65	91.07	80.15	78.8	66.89	60.23	55.84
GloVe 300d (fine-tuned)	75.31	76.2	91.08	80.15	79.0	65.25	58.11	55.48
GloVe 300d + pos encod	72.53	74.34	90.79	78.33	82.4	67.18	59.78	55.12
GloVe 300d + self-attention	74.61	77.74	89.35	79.98	75.4	68.52	61.74	56.48
Dep-based WE 300d	61.01	63.83	83.08	65.36	66.4	66.26	60.76	47.84
Dep-based WE 300d + pos encod	53.68	62.87	78.55	54.7	68.6	65.85	59.11	45.34
Dep-based WE 300d + self-attention	51.68	64.17	80.89	52.88	68.4	67.65	59.6	46.38
Current proposal token embeddings								
SATokE 300d (fixed)	**76.72**	**78.81**	**91.73**	**82.13**	**92.6**	**70.32**	**62.53**	**57.72**
SATokE 300d (fine-tuned)	72.05	71.13	89.85	76.41	75.2	69.44	59.11	53.97

1.6% for MSR, 1.5% for SICK and 0.7% for SNLI-20k. However, the obtained scores are still lower than those using the LSTM architecture on top of the same embeddings or any of the architectures with token embeddings. For sentence classification, results using self-attention are mostly similar to those without.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 10. Comparison of the results obtained using the proposed syntactically aware token embeddings (SATokE 300-dimensional) to results obtained using the 1024-dimensional ELMo token embeddings (Peters *et al*. 2018). Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**

Architecture/Token Embeddings	*'Inappropriate'* datasets				*'Appropriate'* datasets			
	MR	CR	SUBJ	SST-2	TREC	MSR	SICK	SNLI-20k
LSTM								
ELMo 1024d (fixed)	79.24	81.54	**94.27**	84.33	**94.0**	69.10	61.63	**60.64**
ELMo-F 1024d (fine-tuned)	**79.43**	**82.36**	93.77	**84.38**	92.0	68.5	61.85	59.15
SATokE 300d (fixed)	74.39	73.0	91.19	79.6	84.4	**69.27**	**63.59**	56.76
CNN								
ELMo 1024d (fixed)	**78.38**	82.10	93.14	83.01	93.4	66.20	57.88	56.62
ELMo-F 1024d (fine-tuned)	78.03	**82.6**	**93.27**	**84.0**	**92.6**	66.84	60.31	56.4
SATokE 300d (fixed)	76.72	78.81	91.73	82.13	**92.6**	**70.32**	**62.53**	**57.72**

Table 11. Comparison of the results obtained using the proposed syntactically aware token embeddings (SATokE 300-dimensional) to results obtained using the 256-dimensional ELMo token embeddings (Peters *et al*. 2018). Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**

Architecture/token embeddings	*'Inappropriate'* datasets				*'Appropriate'* datasets			
	MR	CR	SUBJ	SST-2	TREC	MSR	SICK	SNLI-20k
LSTM								
ELMo 256d (fixed)	**74.93**	**77.32**	**92.20**	**80.64**	**91.00**	64.00	57.37	42.44
ELMo-F 256d (fine-tuned)	73.72	77.29	91.93	78.06	89.80	64.46	57.78	40.63
SATokE 300d (fixed)	74.39	73.00	91.19	79.60	84.40	**69.27**	**63.59**	**56.76**
CNN								
ELMo 256d (fixed)	74.80	78.38	91.99	80.37	90.6	65.44	50.48	41.96
ELMo-F 256d (fine-tuned)	72.53	77.56	91.14	78.83	89.8	66.43	51.18	40.98
SATokE 300d (fixed)	**76.72**	**78.81**	91.73	**82.13**	**92.6**	**70.32**	**62.53**	**57.72**

5.2 Comparison to other token embeddings

As by construction token embeddings already encode information about the position and role of the token in the sentence, positional encodings and self-attention might bring only marginal additional value. In practice, the situation is even worse as they, in most cases, deteriorate the performance of input embeddings (see Table 9 for Dep-based WE for example). Tables 10 and 11 thus directly compare different token embeddings without positional encodings and self-attention: ELMo 1024-dimensional, ELMo-F 1024-dimensional (fine-tuned version), ELMo 256-dimensional and ELMo-F 256-dimensional (fine-tuned version) to our proposed 300-dimensional embeddings SATokE.

When compared to the 1024-dimensional ELMo embeddings (as in Table 10), on the *'appropriate'* datasets, SATokE surpasses ELMo on the SICK and MSR datasets using an LSTM architecture

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

and on all datasets when input to a CNN. The overall best performance is obtained using SATokE for the MSR and SICK datasets while ELMo yields better results for the TREC and SNLI-20k datasets. This could be somewhat explained by the data statistics presented in Table 5: MSR and SICK are indeed the two datasets that have both the lowest percentage of sentences without SUBJ and without OBJ (2% and 3% for MSR; 6% and 7% for SICK). In contrast to these, SNLI-20k and TREC have considerably higher percentages of sentences without SUBJ and/or OBJ relations (12% and 25% for TREC; 24% and 10% for SNLI-20k), despite a low average sentence length. This may indicate the important role that parsing plays into obtaining meaningful representations through SATokE.

For the *'inappropriate'* datasets, the best results are indeed obtained using ELMo embeddings: fixed for the SUBJ dataset and fine-tuned for the remaining three datasets. However, the difference between results obtained using the fixed and fine-tuned versions of ELMo are non-significant for the MR and SST-2 datasets when using an LSTM and for the MR and SUBJ datasets when using CNN. From an architecture point of view, using a CNN seems to provide the best results for the CR dataset, while all the other best results are obtained with an LSTM.

In general, we believe these results can be explained by looking into the nature and characteristics of the datasets: MSR and SICK have a less diverse vocabulary set and contain syntactically rich structures and well-formed sentences. The last aspect is important when constructing embeddings that rely on accurate parsing. Thus, this could explain the benefit of using our syntactically informed token embeddings on these datasets.

On the other hand, sentences in the *'inappropriate'* datasets have a more varied vocabulary and not always a well-formed syntactic structure: for example, MR, CR and SST-2 are all datasets of reviews. Thus, from a structural point of view, these datasets can be more challenging, as outlined by both the average sentence length and the percentage of sentences without a SUBJ and/or an OBJ dependency relation in Table 5. Given our token embeddings depend on a proper sentence structure, we expect to observe a lower performance on such examples. Additionally, these datasets are more likely to contain typos, interjections or words for which a pre-trained GloVe embedding is not available. Since SATokE relies on both parsing information and the availability of GloVe pre-trained embeddings, these aspects can affect the quality of the obtained SATokE embeddings. In contrast to this, ELMo additionally uses character-level embeddings enabling considerable improvements for the representation of such unknown words. Thus, obtaining better results using token embeddings trained on large amount of data with a language modelling objective and with character-level information is not a surprising result on these *'inappropriate'* datasets.

The results in Table 10 are provided using 1024-dimensional ELMo embeddings, while our proposed SATokE embeddings are 300-dimensional (to enable the compatibility to 300-dimensional GloVe representations on which they are constrained). Thus, to provide a fair comparison to embeddings of similar dimensions, Table 11 compares SATokE to 256-dimensional ELMo token embeddings. The results show that, given similar dimensionality, SATokE outperforms ELMo across all but one dataset (SUBJ), regardless of whether they are labelled as *'appropriate'* or *'innapropriate'* and even despite fine-tuning the ELMo embeddings.

The significant drop in performance observed with respect to Table 10 suggest that the superior results of ELMo 1024d are linked to its increased dimensionality compared to ELMo 256d. This dimensionality allows it to retain useful information. It is important to note that ELMo 256d fails to capture the same amount and quality of information as SATokE, despite it being trained on the same amount of data as ELMo 1024d, which is far larger than the amount of data used in the case of SATokE.

With respect to the architecture type, CNN continues to be preferred for SATokE 300d, except in the case of the SICK dataset when the best results are obtained using an LSTM. For ELMo 256d, the best results are obtained using CNN for the CR and MSR datasets and using LSTM for all the others. Additionally, it can be observed that the difference between scores obtained

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 12. Sentence classification and sentence pair classification results in terms of accuracy (%). Comparison of the proposed syntactically-aware token embeddings (SATokE 300d) to token embeddings computed using the *2-steps* setting (SATokE 300d *2-steps*) and to token embeddings computed using only adjacency information from the sentence graph (SATokE 300d only adjacency). Best results are in **bold**

Architecture/token embeddings	*'Inappropriate'* datasets				*'Appropriate'* datasets			
	MR	CR	SUBJ	SST-2	TREC	MSR	SICK	SNLI-20k
CNN								
SATokE 300d	**76.72**	**78.81**	**91.73**	**82.13**	**92.6**	**70.32**	**62.53**	**57.72**
SATokE 300d *2-steps*	75.83	78.11	91.34	81.85	89.8	70.20	62.47	56.94
SATokE 300d only adjacency	50.67	63.62	50.94	51.45	33.2	66.43	56.50	36.79

by SATokE and ELMo 256d on the *'appropriate'* datasets is higher (between 1.6% and 18%) than that present on the *'inappropriate'* datasets (no more than 2%), suggesting SATokE clearly yields improvements when *'appropriate'* data are provided.

5.3 Linear versus dependency contexts

Table 9 also shows that the word type embeddings trained on dependency contexts (Levy and Goldberg 2014a) yield consistently worse results than GloVe embeddings in our evaluation, with positional encodings providing an even further performance decrease. This leads us to believe dependency information is more valuable for text understanding when complementing GloVe embeddings and when injected into the token representations directly than when used as context for creating generic word embeddings from a larger corpus. This is consistent with Ghannay *et al.* (2016) who showed that embeddings trained on dependency contexts outperform other embeddings on tasks such as NER, POS tagging or chunking, yet obtain lower results on semantic tasks such as analogical reasoning and similarity tasks.

Furthermore, we observe that the difference in performance between GloVe and Dep-based WE is considerably smaller for some datasets than for others: only 1.7% difference in performance for MSR and 1.2% for SICK and between 6.5% and 14% for all the other datasets. Interestingly, the datasets on which word type embeddings trained on dependency contexts are almost as good as GloVe are the same on which our proposed token embeddings outperform ELMo, namely MSR and SICK. We believe this further supports our observations regarding the link between differently constructed embeddings and the characteristics of the sentences in these datasets.

5.4 Token embeddings from the 2-steps set-up

As the best results using our token embeddings for almost all datasets are obtained in the *1-step* setup using a CNN architecture, we only test such an architecture with token embeddings obtained in the *2-steps* set-up. The results using the token embeddings from the *2-steps* setting are slightly lower than those obtained in the *1-step* setting as it can be seen from Table 12: *SATokE 300d 2-steps*. This is an expected result as the difference between the two marks the trade-off between having a complete encoding of the sentences (when relations are learned along with the tokens and thus from the same domain) and having a faster performance (when relations are reused and may come from different domains with potential distribution mismatch). We hypothesize (but do not test) that using a larger training corpus to learn relations embeddings during *(step-1)* of a *2-steps* set-up might improve on the results when using the token embeddings inferred in *(step-2)* for each individual dataset. Leveraging larger corpora to pre-compute relations embeddings is

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 13. Examples of challenging sentence pairs correctly classified using SATokE but misclassified using word type embeddings as input

S1: a dog is chasing another and is holding a stick in its mouth.
S2: a dog is chasing a stick and holding another dog in its mouth.
S1: a baby is licking a dog.
S2: a dog is licking a baby.
S1: a man is holding a sign and is seeking money.
S2: a man is seeking a sign and is holding some money.
S1: the woman is picking up the baby kangaroo.
S2: a kangaroo is picking up the woman's baby.

conceptually similar to how related work uses such large corpora to obtain word type embeddings (Pennington *et al*. 2014; Levy and Goldberg 2014a) that are subsequently used in other tasks (Kim 2014). This makes the volume of the data used for such pre-computations an important aspect of the models. Nevertheless, the results obtained when using the token embeddings from the *2-steps* set-up are still better than results obtained using standard word type embeddings.

5.5 Ablation studies and qualitative analysis

5.5.1 Can syntax help?

We perform an additional experiment by constructing token embeddings leveraging only information from the adjacency graph. We evaluate these token embeddings using the CNN architecture as it has yielded the best results in most of the analysed cases.

The results for all datasets considered are in Table 12: *SATokE 300d only-adjacency*. We can observe the results are consistently worse than all the other results obtained with token embeddings that take into account syntactic information as well. This empirically outlines the importance of using syntax for the creation of token embeddings. While a more in-depth analysis can be performed to look into the individual contribution of different syntactic relations and possibly a different model can be used to better capture adjacency information, the current results demonstrate that the improvement comes from the syntactic information and not only adjacency.

5.5.2 On which examples can we observe improvements using SATokE?

Further, we look at challenging examples in which the SATokE token embeddings perform better than the standard word type embeddings. Some of these examples are presented in Table 13. We notice the sentences on which token embeddings outperform type embeddings tend to be those which require an understanding that goes beyond word surface level. We argue the token embeddings provide a useful representation in these cases. Table 14 outlines examples of sentences taken from the SICK dataset which are correctly classified when using the SATokE token embeddings as input, but misclassified when using the ELMo token embeddings. The sentences in these examples have the particularity of sharing the same vocabulary within a sentence pair, yet having different meanings. The first and third examples in Table 14 represent cases in which the agent and object of an action are reversed; the second example outlines the importance of being able to distinguish the entity performing a particular action, while the last example requires distinguishing on which entity a certain characteristic applies to. Doing well on such examples requires thus not only being able to distinguish between the subject and the object of an action, but also keeping track of other types of dependencies. Thus, syntactically informed representations can be of particular use in such cases.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Table 14. Examples of challenging sentence pairs correctly classified using SATokE but misclassified using the ELMo token embeddings as input

S1: a boy is whacking a man with a sword.
S2: a man is whacking a boy with a sword.
S1: people are singing and a clown is dancing.
S2: a clown is singing and people are dancing.
S1: a dog is teasing a monkey at the zoo.
S2: a monkey is teasing a dog at the zoo.
S1: a small girl is riding in a toy car.
S2: a small toy girl is in a riding car.

Table 15. Examples of most similar words to a given token

Sentence	Nearest types to token	Nearest types to type
things go terribly wrong for the honest **bank** manager [. . .]	banks, banking, credit, central, investment, financial	banks, banking, credit, financial
a tan dog is splashing in the water on the **bank** of a pond	fed, money, branch, shore, wall	
i've read about the issues with the scroll **bar** [. . .]	bars, menu, screen, shop, buttons, camera	bars, cafe, pub, lounge, room, shop
[...] first cartoon to look as if it were being shown on the unknown_nn television screen of a sports **bar**.	bars, cafe, lounge, hostess, pub	
if so, then this movie will **touch** your soul. . .	touches, senses, lightness, sweetness	touches, hand, screen
we are fully prepared to roll out the [**touch** - screen] machines for the 2004 presidential primary [. . .]	touches, buttons, handheld, screen, click, devices	

Finally, we look at specific examples of sentences with polysemous words. Such examples are shown in Table 15. For each sentence, we find the polysemous word (in bold) and we further provide a list of the most similar word types (by cosine similarity) to the token representation of the searched word. For comparison, we also provide the list of most similar word types to the type representation of the same word (namely its GloVe embedding). We can observe the token embeddings manage to capture meaning and distinguish between the senses of a word, although they have not been explicitly trained on a word sense disambiguation task. This is another clue that the produced token embeddings encode meaningful local context information obtained from the graph.

6. Conclusion

In this work, we propose a method to encode arbitrary graphs of linguistic structure and produce context-aware token embeddings using a tensor factorisation approach. The proposed method provides a flexible framework that allows for the explicit incorporation of additional knowledge into token representations. The obtained token embeddings encode information about the context

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

they appear in as well as about the structure of the sentence. We evaluated these tokens on a series of binary and multi-class classification tasks by integrating them as input features to two families of neural network models frequently used in the literature. We demonstrated that the computed token embeddings constitute an improved choice of features in both architectures, compared to word type embeddings, even when the latter are learned using contexts based on syntactic dependencies. We also showed that, depending on the nature of the dataset and the characteristics of the sentences within it, our token embeddings can be more appropriate than alternatives in the literature: results using the SATokE embeddings surpass those obtained with the best ELMo 1024d token embeddings across all *'appropriate'* datasets when using a CNN architecture and provide in general competitive results using an LSTM architecture. Furthermore, it can be observed that, when compared to the ELMo 256d token embeddings that have comparable dimensionality, SATokE obtains better performance due to a significance drop in performance of the ELMo token embeddings for most datasets. An ablation study outlined the importance of using syntactic information in the creation of the token embeddings and the conducted qualitative analysis showed promising results for challenging sentence understanding scenarios. In future work, we consider investigating the impact of relations provided by a semantic parser and the use of our method in the context of semi-supervised learning.

References

Bahdanau D., Cho, K. and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2014 International Conference on Learning Representations*.

Bansal M., Gimpel K. and Livescu, K. (2014). Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Baroni M. and Lenci A. (2010). Distributional memory: a general framework for corpus-based semantics. *Journal of Computational Linguistics* **36**(4), 673–721.

Bengio Y., Ducharme R., Vincent P. and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research* **3**, 1137–1155.

Bentivogli L., Bernardi R., Marelli M., Menini S., Baroni M. and Zamparelli R. (2016). Sick through the semeval glasses. lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Journal of Language Resources and Evaluation* **50**, 95–124.

Blacoe W. and Lapata M. (2012). A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Bordes A., Usunier N., Garcia-Duran A., Weston J. and Yakhnenko O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*. Curran Associates Inc., pp. 2787–2795.

Bowman S.R., Angeli G., Potts C. and Manning C.D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Chen X., Liu Z. and Sun M. (2014). A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K. and Kuksa P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research* **12**, 2493–2537.

Conneau A. and Kiela D. (2018). Senteval: an evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Conneau A., Kiela D., Schwenk H., Barrault L. and Bordes A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Dasigi P., Ammar W., Dyer C. and Hovy E.H. (2017). Ontology-aware token embeddings for prepositional phrase attachment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Devlin J., Chang M.-W., Lee, K. and Toutanova K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dolan B. and Brockett C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing*.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Faruqui M., **Dodge J.**, **Jauhar S.K.**, **Dyer C.**, **Hovy E.H. and Smith N.A.** (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Gehring J., **Auli M.**, **Grangier D.**, **Yarats D. and Dauphin Y.** (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*.

Ghannay S., **Favre B.**, **Estève Y. and Camelin N.** (2016). Word embedding evaluation and combination. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia, pp. 300–305. European Language Resources Association (ELRA).

Grefenstette E. and Sadrzadeh M. (2011). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Grefenstette G. (1994). *Explorations in Automatic Thesaurus Discovery*. USA: Kluwer Academic Publishers.

Henderson J. (2003). Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Honnibal M. and Johnson M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Hochreiter S. and Schmidhuber J. (1997). Long short-term memory. *Neural Computing* **9**(8), 1735–1780.

Hu M. and Liu B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

İrsoy O. and Cardie C. (2014). Deep recursive neural networks for compositionality in language. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*.

Kalchbrenner N., **Grefenstette E. and Blunsom P.** (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Kim Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Kingma D. and Ba J. (2014). Adam: a method for stochastic optimization. In *Proceedings of the 2014 International Conference on Learning Representations*.

Kiros R., **Zhu Y.**, **Salakhutdinov R.R.**, **Zemel R.**, **Urtasun R.**, **Torralba A. and Fidler S.** (2015). Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*.

Levy O. and Goldberg Y. (2014a). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Levy O. and Goldberg Y. (2014b). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*. MIT Press, pp. 2177–2185.

Li X. and Roth D. (2002). Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*.

Lin D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*.

Liu P., **Qiu X. and Huang X.** (2015). Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of the 24th International Conference on Artificial Intelligence*.

Liu Y., **Sun C.**, **Lin L. and Wang X.** (2016). Learning natural language inference using bidirectional lstm model and inner-attention. CoRR abs/1605.09090.

Marcus M.P., **Marcinkiewicz M.A.** & **Santorini B.** (1993). Building a large annotated corpus of english: the penn treebank. Journal of Computational Linguistics - Special Issue on Using Large Corpora.

McCann B., **Bradbury J.**, **Xiong C.** & **Socher R.** (2017). Learned in translation: contextualized word vectors. In *Advances in Neural Information Processing Systems 30*, pp. 6294–6305.

Melamud O., **Goldberger J. and Dagan I.** (2016). context2vec: learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Mikolov T., **Chen K.**, **Corrado G.S.** & **Dean J.** (2013a). Efficient estimation of word representations in vector space. CoRR abs/1301.3781.

Mikolov T., **Sutskever I.**, **Chen K.**, **Corrado G.S.** & **Dean**, **J.** (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., pp. 3111–3119.

Mitchell J. and Lapata M. (2010). Composition in distributional models of semantics. Journal of Cognitive Science **34**, 1388–1429.

Mrkšic N., **OSéaghdha D.**, **Thomson B.**, **Gašic M.**, **Rojas-Barahona L.**, **Su P.-H.**, **Vandyke D.**, **Wen T.-H.** & **Young S.** (2016). Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Neelakantan A., **Shankar J.**, **Passos A. and McCallum A.** (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

https://doi.org/10.1017/S1351324920000297 Published online by Cambridge University Press

Nickel M., **Tresp V. and Kriegel H.-P.** (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*. Omnipress, pp. 809–816.

Padó S. and Lapata M. (2007). Dependency-based construction of semantic space models. *Journal of Computational Linguistics* **33**, 161–199.

Pang B. and Lee L. (2004). A sentimental education: sentiment analysis using subjectivity. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Pang B. and Lee L. (2005). Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Pennington J., **Socher R. and Manning C.D.** (2014). Glove: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Peters M., **Ammar W.**, **Bhagavatula C. and Power R.** (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Peters M.E., **Neumann M.**, **Iyyer M.**, **Gardner M.**, **Clark C.**, **Lee K. and Zettlemoyer L.** (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Salant S. and Berant J. (2018). Contextualized word representations for reading comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Socher R., **Bauer J.**, **Manning C.D**, **and Ng A.Y.** (2013a). Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Socher R., **Huang E.H.**, **Pennington J.**, **Ng A.Y. and Manning C.D.** (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*.

Socher R., **Huval B.**, **Manning C.D. and Ng A.Y.** (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Socher R., **Manning C.D. and Ng A.Y.** (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Socher R., **Perelygin A.**, **Wu J.**, **Chuang J.**, **Manning C.D.**, **Ng A.Y. and Potts C.** (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Tang J., **Qu M. and Mei Q.** (2015). Pte: predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Trouillon T., **Welbl J.**, **Riedel S.**, **Gaussier É. and Bouchard G.** (2016). Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*.

Tu L., **Gimpel K. and Livescu K.** (2017). Learning to embed words in context for syntactic tasks. CoRR abs/1706.02807.

Vaswani A., **Shazeer N.**, **Parmar N.**, **Uszkoreit J.**, **Jones L.**, **Gomez A.N.**, Kaiser . and **Polosukhin I.** (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 5998–6008.

Vulic I., **Mrksic N.**, **Reichart R.**, **Séaghdha D.Ó.**, **Young S.J. and Korhonen A.** (2017). Morph-fitting: fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Weir D., **Weeds J.**, **Reffin J. and Kober T.** (2016). Aligning packed dependency trees: a theory of composition for distributional semantics. *Journal of Computational Linguistics* **42**, 727–761.

Westera M. & **Boleda G.** (2019). Don't blame distributional semantics it can't do entailment. In *Proceedings of the 13th International Conference on Computational Semantics*.

Zhao H., **Lu Z.** & **Poupart P.** (2015). Self-adaptive hierarchical sentence model. In *Proceedings of the 24th International Conference on Artificial Intelligence*.

Zou W.Y., **Socher R.**, **Cer D. and Manning C.D.** (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Cite this article: Popa DN, Perez J, Henderson J and Gaussier E (2021). Towards syntax-aware token embeddings. *Natural Language Engineering* **27**, 691–720. https://doi.org/10.1017/S1351324920000297