

ISSN 2499-4553

IJCoL

Italian Journal
of Computational Linguistics

Rivista Italiana
di Linguistica Computazionale

Volume 5, Number 1
june 2019

aA ccademia
university
press

editors in chief

Roberto Basili

Università degli Studi di Roma Tor Vergata

Simonetta Montemagni

Istituto di Linguistica Computazionale “Antonio Zampolli” - CNR

advisory board

Giuseppe Attardi

Università degli Studi di Pisa (Italy)

Nicoletta Calzolari

Istituto di Linguistica Computazionale “Antonio Zampolli” - CNR (Italy)

Nick Campbell

Trinity College Dublin (Ireland)

Piero Cosi

Istituto di Scienze e Tecnologie della Cognizione - CNR (Italy)

Giacomo Ferrari

Università degli Studi del Piemonte Orientale (Italy)

Eduard Hovy

Carnegie Mellon University (USA)

Paola Merlo

Université de Genève (Switzerland)

John Nerbonne

University of Groningen (The Netherlands)

Joakim Nivre

Uppsala University (Sweden)

Maria Teresa Pazienza

Università degli Studi di Roma Tor Vergata (Italy)

Hinrich Schütze

University of Munich (Germany)

Marc Steedman

University of Edinburgh (United Kingdom)

Oliviero Stock

Fondazione Bruno Kessler, Trento (Italy)

Jun-ichi Tsujii

Artificial Intelligence Research Center, Tokyo (Japan)

Cristina Bosco

Università degli Studi di Torino (Italy)

Franco Cutugno

Università degli Studi di Napoli (Italy)

Felice Dell'Orletta

Istituto di Linguistica Computazionale "Antonio Zampolli" - CNR (Italy)

Rodolfo Delmonte

Università degli Studi di Venezia (Italy)

Marcello Federico

Fondazione Bruno Kessler, Trento (Italy)

Alessandro Lenci

Università degli Studi di Pisa (Italy)

Bernardo Magnini

Fondazione Bruno Kessler, Trento (Italy)

Johanna Monti

Università degli Studi di Sassari (Italy)

Alessandro Moschitti

Università degli Studi di Trento (Italy)

Roberto Navigli

Università degli Studi di Roma "La Sapienza" (Italy)

Malvina Nissim

University of Groningen (The Netherlands)

Roberto Pieraccini

Jibo, Inc., Redwood City, CA, and Boston, MA (USA)

Vito Pirrelli

Istituto di Linguistica Computazionale "Antonio Zampolli" - CNR (Italy)

Giorgio Satta

Università degli Studi di Padova (Italy)

Gianni Semeraro

Università degli Studi di Bari (Italy)

Carlo Strapparava

Fondazione Bruno Kessler, Trento (Italy)

Fabio Tamburini

Università degli Studi di Bologna (Italy)

Paola Velardi

Università degli Studi di Roma "La Sapienza" (Italy)

Guido Vetere

Centro Studi Avanzati IBM Italia (Italy)

Fabio Massimo Zanzotto

Università degli Studi di Roma Tor Vergata (Italy)

Danilo Croce

Università degli Studi di Roma Tor Vergata

Sara Goggi

Istituto di Linguistica Computazionale "Antonio Zampolli" - CNR

Manuela Speranza

Fondazione Bruno Kessler, Trento

Registrazione presso il Tribunale di Trento n. 14/16 del 6 luglio 2016

Rivista Semestrale dell'Associazione Italiana di Linguistica Computazionale (AILC)
© 2018 Associazione Italiana di Linguistica Computazionale (AILC)



Associazione Italiana di
Linguistica Computazionale

direttore responsabile
Michele Arnese

Pubblicazione resa disponibile
nei termini della licenza Creative Commons
Attribuzione – Non commerciale – Non opere derivate 4.0



isbn 978-88-31978-89-7

Accademia University Press
via Carlo Alberto 55
I-10123 Torino
info@aAccademia.it
www.aAccademia.it/IJCoL_5_1



Accademia University Press è un marchio registrato di proprietà
di LEXIS Compagnia Editoriale in Torino srl

CONTENTS

Nota Editoriale <i>Roberto Basili, Simonetta Montemagni</i>	7
On the Readability of Kernel-based Deep Learning Models in Semantic Role Labeling Tasks over Multiple Languages <i>Daniele Rossini, Danilo Croce, Roberto Basili</i>	11
State-of-the-art Italian dependency parsers based on neural and ensemble systems <i>Oronzo Antonelli, Fabio Tamburini</i>	33
Negated Adjectives and Antonyms in Distributional Semantics: not similar? <i>Laura Aina, Raffaella Bernardi, Raquel Fernández</i>	57
Event Knowledge in Compositional Distributional Semantics <i>Ludovica Pannitto, Alessandro Lenci</i>	73
Multi-source Transformer for Automatic Post-Editing of Machine Translation Output <i>Amirhossein Tebbifakhr, Matteo Negri, Marco Turchi</i>	89

Emerging Topics from the Fifth Italian Conference on Computational Linguistics

Roberto Basili*

Università di Roma, Tor Vergata

Simonetta Montemagni**

ILC - CNR

Il primo numero del quinto anno della rivista *Italian Journal of Computational Linguistics (IJCoL)*, la rivista italiana promossa dall'Associazione Italiana di Linguistica Computazionale (AILC - www.ai-lc.it), è un volume miscelaneo i cui articoli documentano lavori di ricerca risultati particolarmente promettenti nell'ambito della Conferenza *CLiC-it 2018* (Torino, 10-12 dicembre 2018) insieme ad un contributo originale sottomesso per la pubblicazione sulla rivista. La selezione di contributi qui rappresentata documenta la ricerca condotta in diversi ambiti, che spaziano da studi sulle applicazioni di modelli neurali all'apprendimento in sistemi di Natural Language Processing, sui modelli vettoriali della semantica distribuzionale sino a metodi complessi di inferenza linguistica, utili in applicazioni moderne, quali la traduzione automatica. I temi affrontati dunque coprono sviluppi recenti e molto fecondi della ricerca in linguistica computazionale, come ad esempio la trasparenza epistemologica dei modelli induttivi basati su reti neurali, l'annotazione sintattica automatica dell'italiano, l'utilizzo di tecniche avanzate di apprendimento neurale, quali gli algoritmi basati sui "transformer", per arrivare ai modelli distribuzionali del significato.

Gli articoli sui temi emergenti sono stati, come sempre, selezionati attraverso un processo iterativo di *peer-review*: come contributo alla conferenza e come candidato ai premi di "Best Young Paper" e "Distinguished Young Paper". Infine, la versione estesa degli articoli selezionati da *CLiC-it 2018* e gli articoli proposti per la pubblicazione su *IJCoL* sono stati oggetto di revisione paritaria come articolo di rivista scientifica.

Nel contributo di Rossini, Croce e Basili viene discusso il problema della leggibilità dei modelli neurali del linguaggio naturale, che rappresenta una questione centrale quando si considera l'impatto di tali modelli all'interno di scenari applicativi del mondo reale. In particolare, viene proposta una tecnica di *embedding* basata su un metodo matematico per la rappresentazione di spazi caratterizzati da un numero elevato di dimensioni (denominata *metodologia di Nyström*), finalizzata a supportare inferenze analogiche relative alla relazione causale tra i dati di addestramento e le decisioni della rete neurale. I metodi *kernel* utilizzati per costruire gli *embeddings* di *Nyström* sono in grado di catturare informazione semantica, sintattica e lessicale, fornendo spiegazioni fondate e percepite come naturali dall'utente. La valutazione quantitativa riportata si riferisce a un compito di inferenza semantica, il *Semantic Role Labeling*, condotto per l'italiano e l'inglese qui usato come caso di studio. I risultati raggiunti mostrano che le spiegazioni basate su *kernel* semantici sono particolarmente affidabili ed efficaci, consentendo alle

* Dept. of Enterprise Engineering - Via del Politecnico 1, 00133 Roma
E-mail: basili@info.uniroma2.it

** Istituto di Linguistica Computazionale "A. Zampolli", CNR - Via Moruzzi 1, 56124 Pisa
E-mail: simonetta.montemagni@ilc.cnr.it

strutture linguistiche di contribuire in modo significativo alla percezione dell'utente riguardo all'affidabilità della decisione.

Nel secondo articolo, Antonelli e Tamburini riportano i risultati di una vasta sperimentazione condotta con architetture neurali per l'analisi sintattica a dipendenze di testi italiani. Gli esperimenti fanno uso di due *treebanks*, rappresentative di diversi generi testuali, sintatticamente annotate secondo lo schema delle *Universal Dependencies*, ad oggi uno standard "de facto" per l'annotazione sintattica a dipendenze. Viene proposta una nuova architettura di analisi basata su sistemi che combinano il risultato di diversi analizzatori sintattici a dipendenze (*ensemble systems*), che differisce dalle precedenti tecniche di combinazione sia per i parsers utilizzati sia per il metodo di combinazione degli output. Con questo studio è stato possibile valutare l'impatto di diversi algoritmi e tecniche di combinazione degli output rispetto sia a misure standard di valutazione sia all'analisi della qualità dell'albero sintattico generato. I risultati ottimali sono stati ottenuti applicando lo schema di voto a maggioranza, nonostante il fatto che tale metodo non garantisca che l'albero generato sia ben formato.

Seguono due articoli che condividono l'approccio distribuzionale all'analisi semantica. Il contributo di Aina, Bernardi e Fernández riporta i risultati di uno studio di semantica distribuzionale dedicato alla modellazione del fenomeno della negazione di un aggettivo, analizzato in rapporto alla corrispondente coppia di antonimi per la lingua inglese (es. *not cold* vs *cold - hot*). Dopo aver costruito la rappresentazione vettoriale di un insieme di antonimi e delle relative negazioni a partire dalla loro distribuzione nei contesti linguistici in cui ricorrono, l'analisi ha studiato le similarità contestuali osservabili riguardo agli aggettivi oggetto di negazione e gli aggettivi della corrispondente coppia di antonimi. I risultati raggiunti dimostrano che la negazione di un aggettivo (es. *not cold*) è maggiormente simile in termini distribuzionali all'aggettivo stesso che al suo antonimo (ovvero a *cold* piuttosto che a *hot*). L'analisi è stata condotta in relazione a diversi tipi di coppie di antonimi, ovvero *antonimi morfologici* che condividono la radice lessicale, e *antonimi lessicali* contraddistinti da radici lessicali diverse (all'interno dei quali si distinguono antonimi *contrari* vs *contraddittori*). Mentre nel caso di antonimi morfologici la maggiore similarità distribuzionale dell'aggettivo negato rispetto all'aggettivo stesso risulta meno marcata, non sono emerse differenze di comportamento significative tra i diversi tipi di antonimi lessicali e di negazione (semplice vs doppia).

Il contributo di Pannitto e Lenci si pone come obiettivo la valutazione del contributo dell'informazione eventiva nel processo di comprensione della frase. Partendo dalla constatazione che la maggior parte dei modelli proposti nell'ambito della semantica distribuzionale compositiva si basa sull'utilizzo dei soli vettori lessicali, gli autori propongono di arricchire il miglior modello presente in letteratura, costituito dalla somma di vettori (assunta come *baseline*), con informazione distribuzionale sulla conoscenza degli eventi attivata dalle parole nella frase. Recenti studi in ambito psicolinguistico mostrano che il lessico attiva conoscenza sulla struttura degli eventi e sui loro partecipanti prototipici. I risultati degli esperimenti confermano che i vettori distribuzionali non contengono informazione eventiva sufficiente e che, in linea con l'evidenza psicolinguistica, la conoscenza attivata dal lessico svolge un ruolo centrale nella costruzione della rappresentazione semantica della frase. L'approccio proposto è stato valutato su due risorse, RELPRON e un insieme di frasi transitive annotate sintatticamente e semanticamente, con risultati promettenti che mostrano un netto e sistematico miglioramento rispetto ai risultati della baseline, confermando così l'ipotesi di partenza sul ruolo dell'informazione eventiva attivata dal lessico nel processo di comprensione della frase.

Chiude il volume l'articolo di Tebbifakhr e colleghi che propone un nuovo approccio alle fasi di *post-editing* automatico (APE) di un sistema di traduzione automatica neurale. I metodi APE operano sull'output del sistema di traduzione automatica considerando anche la corrispondente frase di origine. Nel contributo gli autori propongono un'architettura neurale basata su *transformer* per APE, i cui vantaggi computazionali e ingegneristici sono significativi, in quanto possono essere facilmente parallelizzati e mantenuti nel tempo. Nell'articolo è anche indagata l'ipotesi relativa al ruolo della quantità dei dati usati per l'addestramento. Relativamente a ciò viene dimostrato che insiemi di dati annotati di grandi dimensioni non implicano sempre prestazioni migliori, poiché l'addestramento del sistema sulla base di piccoli e mirati sottoinsiemi di dati del dominio può portare a una precisione maggiore.

Questa breve sintesi non fa giustizia ai molti stimoli forniti dai lavori del presente volume e alla loro variegata rassegna di metodologie e applicazioni innovative nel trattamento automatico della lingua. Lasciamo, come sempre, al lettore il piacere di approfondirli direttamente nell'interesse di questo volume.

Editorial Note Summary

It is with great pleasure that we introduce the first volume of the fifth year of the *Italian Journal of Computational Linguistics* (IJCoL) promoted by the *Associazione Italiana di Linguistica Computazionale* (AILC - www.ai-lc.it). This is a miscellaneous volume which collects a selection of particularly promising research contributions inspired by young researchers at the CLiC-it 2018 Conference (Turin, 10-12 December 2018). This selection of contributions from CLiC-it 2018 focuses on different paradigms and research aspects, ranging from neural learning for NLP to distributional semantics and applications such as machine translation. As for the other miscellaneous issues, the papers have been selected through an iterative peer-review process. Each selected conference article underwent the evaluation as a contribution to the conference and as a candidate for the "Best Young Paper" and "Distinguished Young Paper" awards of CLiC-it 2018. Finally, both extended and newly submitted papers have been peer-reviewed as journal articles.

The paper by Rossini, Croce and Basili discusses the problem of the readability of neural natural language models, a relevant research issue, as for its impact in real world NLP applications. An embedding technique (based on a mathematical method to represent large dimensional feature spaces, called the *Nyström* methodology) is here proposed as a way to support analogy-driven inferences about the causal relationship between training data and a neural net decisions. Kernel methods used to build the Nyström embedding capture grammatical and lexical semantic information, making explanations meaningful and naturally perceived by the user. The reported quantitative evaluation uses a semantic inference task, i.e. Semantic Role Labeling, both in English and Italian, as a reference test bed. Results suggest that explanations based on such kernels let semantic and syntagmatic structures to contribute to convincing arguments. They are in fact rather effective in the user assessment about the correctness of machine decisions.

In the second paper, Antonelli and Tamburini present an extensive experimentation on neural architectures for parsing Italian texts. The experiments use two Italian treebanks with different text types annotated according to the Universal Dependencies project standards. A new parsing architecture based on ensemble systems is proposed, extending previous ensemble techniques by a voting capability based on individual dependency relations. The aim has been to assess the impact of different algorithms and ensemble techniques with respect to standard parsing quality measures as well as

to the linguistic quality of the resulting parse tree. Optimal results are obtained when the majority-voting ensemble scheme is applied, whose major drawback is the fact that it does not always ensure the well-formedness of the generated dependency tree.

Two articles follow that share the distributional approach to semantic analysis. The contribution by Aina, Bernardi and Fernández reports the results of a distributional semantics study devoted to the modeling of the negation of an adjective analyzed in relation to the corresponding pair of antonyms for the English language (eg *emph not cold* vs *emph cold* - *emph hot*). After having constructed the vector representation of a set of antonyms and their negations starting from their distribution in the linguistic contexts in which they occur, the analysis has concerned the similarities found between the contexts of the adjectives object of negation and the adjectives of the corresponding pair of antonyms. The results achieved show that the negation of an adjective (e.g. *not cold*) is more similar in distributional terms to the adjective itself than to its antonym (i.e. to *cold* rather than to *hot*). The analysis was carried out in relation to different types of pairs of antonyms, namely *morphological antonyms* sharing the lexical root, and *lexical antonyms* characterized by different lexical roots (within which *contrary* vs *contradictory* antonyms are distinguished). While in the case of morphological antonyms the greater distributional similarity of the negated adjective with respect to the adjective itself is less marked, no significant differences emerged between the different types of lexical antonyms and negation (simple vs double).

The paper by Pannitto and Lenci is aimed at evaluating the contribution of event information in the process of sentence comprehension. Starting from the observation that most of the models proposed in the compositional distributional semantics literature are based on the use of lexical vectors only, the authors propose to enrich the best performing model in the literature, consisting in vector addition (assumed as baseline), with distributional information about events which is activated by the words in the sentence. Recent studies in the psycholinguistic literature show that the lexicon activates the knowledge about the structure of events and their prototypical participants. The results of the experiments presented in the paper confirm that distributional vectors do not contain sufficient event information and that, in line with psycholinguistic evidence, the knowledge activated by the lexicon plays a central role in the construction of the semantic representation of the sentence. The proposed approach has been evaluated against two resources, RELPRON and a set of transitive sentences annotated syntactically and semantically, with promising results showing a clear and systematic improvement with respect to the selected baseline. This confirms the hypothesis about the role played by event information which is activated by the lexicon in the sentence comprehension process.

The volume closes with the paper by Tebbifakhr and colleagues proposing a novel approach to the automatic post-editing stages (APE) of a neural machine translation system. APE methods are competitive as they correct the raw translation output by also considering the corresponding source sentence. In the paper the authors propose a Transformer based neural architecture for APE, whose computational and engineering advantages are significant as they can be easily parallelized and maintained. In the paper assumptions about the role of the size of the training dataset are also validated. It is shown that the widely assumed assumption suggesting that larger data sets imply better performances does not always hold, as fine-tuning the system on small in-domain data can yield higher accuracy.

This synthetic view does not exhaust the suggestions and nuances inspired by the papers here collected. We leave the reader the pleasure to discover them through a thoughtful sailing across the rest of the volume contents.

On the Readability of Kernel-based Deep Learning Models in Semantic Role Labeling Tasks over Multiple Languages

Daniele Rossini*

Università di Roma, Tor Vergata

Danilo Croce**

Università di Roma, Tor Vergata

Roberto Basili†

Università di Roma, Tor Vergata

Sentence embeddings are effective input vectors for the neural learning of a number of inferences about content and meaning. Unfortunately, most of such decision processes are epistemologically opaque as for the limited interpretability of the acquired neural models based on the involved embeddings. In this paper, we concentrate on the readability of neural models, discussing an embedding technique (the Nyström methodology) that corresponds to the reconstruction of a sentence in a kernel space, capturing grammatical and lexical semantic information. From this method, we build a Kernel-based Deep Architecture that is characterized by inherently high interpretability properties, as the proposed embedding is derived from examples, i.e., landmarks, that are both human readable and labeled. Its integration with an explanation methodology, the Layer-wise Relevance Propagation, supports here the automatic compilation of argumentations for the Kernel-based Deep Architecture decisions, expressed in form of analogy with activated landmarks. Quantitative evaluation against the Semantic Role Labeling task, both in English and Italian, suggests that explanations based on semantic and syntagmatic structures are rich and characterize convincing arguments, as they effectively help the user in assessing whether or not to trust the machine decisions.

1. Introduction

Nonlinear methods such as Deep Neural Networks achieve state-of-the-art performances in several semantic Natural Language Processing (NLP) tasks (Goldberg 2016; Collobert et al. 2011). The wide spread of Deep Learning is supported by the impressive results and their feature learning capability (Bengio, Courville, and Vincent 2013; Kim 2014): input words and sentences are usually modeled as dense embeddings (i.e., vectors or tensors), whose dimensions correspond to latent semantic concepts acquired during an unsupervised pre-training stage. In similarity estimation, classification, emotional characterization of sentences as well as pragmatic tasks, such as question answering or dialogue, they largely demonstrated their effectiveness to model semantics.

* Dept. of Enterprise Engineering - Via del Politecnico 1, 00133 Rome, Italy.
E-mail: rossini.danie@gmail.com

** Dept. of Enterprise Engineering - Via del Politecnico 1, 00133 Rome, Italy.
E-mail: croce@info.uniroma2.it

† Dept. of Enterprise Engineering - Via del Politecnico 1, 00133 Rome, Italy.
E-mail: basili@info.uniroma2.it

Unfortunately, several drawbacks arise. First, most of the above approaches are epistemologically opaque as for the limited interpretability of the acquired neural models based on the involved embeddings. Second, injecting linguistic information into a Neural Network (NN) without degrading its transparency properties is still a problem with much room for improvement. Word embeddings are widely adopted as an effective pre-training approach, although there is no general agreement about how to provide deeper linguistic information to the NN. Some structured NN models have been proposed (Hochreiter and Schmidhuber 1997; Socher et al. 2013), although usually tailored to specific problems. Recursive NNs (Socher et al. 2013) have been shown to learn dense feature representations of the nodes in a structure, thus exploiting similarities between nodes and sub-trees. Also, Long-Short Term Memory networks (Hochreiter and Schmidhuber 1997) build intermediate representations of sequences, resulting in similarity estimates over sequences and their inner sub-sequences. However, such intermediate representations are strongly task dependent: this is beneficial from an engineering standpoint, but certainly controversial from a linguistic and cognitive point of view. In recent years, many approaches proposed extensions to the previous methods. Semi-supervised models within the multi-task learning paradigm have been investigated (Collobert et al. 2011). Context-aware dense representations (Pennington, Socher, and Manning 2014) and deep representations based on sub-words or characters (Peters et al. 2018; Devlin et al. 2019) successfully model syntactic and semantic information. Linguistically-informed mechanisms have been proposed to train the self-attention to attend syntactic information in a sentence, granting state-of-the-art results in Semantic Role Labeling (Strubell et al. 2018). However, in such approaches, the captured linguistic properties are never made explicit and the complexity of learned latent spaces only exacerbates the interpretability problem. Hence, despite state-of-the-art performances, the complexity of such approaches exacerbates the issue of a straightforward understanding of the linguistic aspects that are responsible for a network decisions. Attempts to solve the interpretability problem of NNs have been proposed in computer vision (Erhan, Courville, and Bengio 2010; Bach et al. 2015), but their extension to the NLP scenario is not straightforward.

We think that any effective approach to meaning representation should be at least epistemologically coherent, that is readable and justified through an argument theoretic lens on interpretation. This means that inferences based on vector embeddings should also naturally correspond to a clear and uncontroversial logical counterpart: in particular, neurally trained semantic inferences should be also epistemologically transparent. In other words, neural embeddings should support model readability, that is to trace back *causal connections* between the implicitly expressed linguistic properties of an input instance and the classification output produced by a model. Meaning representation should thus strictly support the (neural) learning of epistemologically well-founded models.

A possible solution is to provide *explicit information* regarding semantics by relying on linguistic properties of sentences, i.e., by modeling the lexical, syntactic and semantic constraints implicitly encoded in the linguistic structure. That is achieved by learning methods based on tree kernels (Shawe-Taylor and Cristianini 2004; Moschitti 2012; Collins and Duffy 2002a) as the feature space they capture reflects linguistic patterns. Approximation method can then be used to successfully map tree structures into dense vector representations useful to train a neural network. As suggested in (Croce et al. 2017), the Nyström dimensionality reduction method (Williams and Seeger 2001) is of particular interest as it allows to reconstruct a low-dimensional embeddings of the rich kernel space by computing kernel similarities between input examples and a set

of selected instances, called *landmarks*. If methods such as Nyström’s are used over rich Tree Kernels (TKs), the projection vectors will encode information captured by such kernels, which have been proved to account for syntactic as well as semantic evidence (Croce, Moschitti, and Basili 2011). The resulting vectors can be then used as input of an effective neural learner, namely a *Kernel-based Deep Architecture* (KDA), which has been shown to achieve state-of-the-art results in different semantic tasks, such as question classification and semantic role labeling, and naturally favours the generation of explanations for its decisions: this is obtained by integrating it with a model of the activation state of a network, called *Layer-wise Relevance Propagation* (LRP), that traces back the contribution of input layers (and nodes) to the fired output. Such input components correspond, in a KDA, to landmarks, that are real and labeled examples. Thus it is possible to compile argumentations in favor or against its inference: each decision is in fact justified via an analogy with landmarks most linguistically related to the input instance. For example, consider a Question Classification (QA) (Li and Roth 2006) task and the question: “*What year did Oklahoma become a state ?*”, in which the information request is clearly a `NUMBER`. An argument supporting such claim can be constructed by providing an analogy that highlights the linguistic properties which are relevant for the task at hand. E.g.,

Example 1

I think “What year did Oklahoma become a state ?” refers to a `NUMBER` since it recalls me of “The film Jaws was made in what year ?”

A speaker presented with this argument would implicitly detect the important properties shared between the target example and the one used as comparison, e.g., the syntagma “*what year*”, and implicitly evaluate both the quality of the explanation and the trustfulness of the claim (i.e., the classification output) according to the ease of his properties-detection process. In fact, consider an alternative analogy:

Example 2

I think “What year did Oklahoma become a state ?” refers to a `NUMBER` since it recalls me of “What is the population of Mozambique ?”

While both arguments in Example 1 and 2 are supporting a correct claim (i.e., the question refers to a numeric quantity), the second is clearly less convincing as it is harder for a human to identify the connections between the two questions (which, in this case, is due to the fact that, on a finer grain, they refers to two different sub-categories of information).

In this paper, we extend such readability-enhancing approach, proposed in (Croce, Rossini, and Basili 2019), by conducting further experimental investigations on a Semantic Role Labeling task over English and Italian, in order to test its effectiveness in multiple languages. Quantitative evaluation of these outcomes shows that richer explanations based on semantic and syntagmatic structures characterize convincing arguments, in the sense that they provide right assistance to the user in accepting or rejecting the system output. This confirms the epistemological benefit that Nyström embeddings may bring, as linguistically rich and meaningful representations of useful causal connections in a variety of inference tasks.

We first survey approaches to improve the transparency of neural models in Section 2. In Section 3, we present the role of linguistic similarity principles as they are expressed by Semantic Kernels as well as an approximation technique, the Nyström method, to

derive a low-rank matrix reconstructing the input space induced by a Semantic Kernel. The KDA is illustrated in Section 4.1 while the LRP is described in Section 4.2. Section 5 is dedicated to formalizing models that take as input the result of the LRP and are able to generate explanations for the KDA decisions by exploiting its inherent transparency properties, while in Section 5.1 a method for the quantitative evaluation of explanations is defined. The overall system is evaluated against the Argument Classification (AC) step in the Semantic Role Labeling (SRL) chain (Palmer, Gildea, and Xue 2010), in two different languages: English and Italian. Results are discussed in Section 6. Finally, Section 7 summarizes achievements, open issues and future directions of this work.

2. Related work on Interpretability

Advancements of Deep Learning are allowing the exploitation of data-driven models into areas that have profound impacts on society, as health care services, criminal justice systems and financial markets. Consequently, the traditional criticism of epistemological opaqueness of AI-based systems has recently drawn much attention from the research community, as the ability for humans to understand models and suitably weight the assistance they provide is a central issue for the correct adoption of such systems. However, to empower neural models with interpretability properties is still an open problem as it even lacks a broad consensus on the definitions of interpretability and explanation.

(Lipton 2018) analyzed definitions of interpretability and transparency found in literature and structured them across two main dimensions: *Model Transparency*, i.e., understanding the mechanism by which the model works, and *Post-Hoc Explainability* (or *Model Functionality*), i.e., the property by which the system conveys to its users information useful to justify its functioning such as intuitive evidences supporting the output decisions. The latter can be further divided into *global* explanations, i.e., a description of the full mapping the network has learned, and *local* explanations, i.e., motivations underlying a single output. Examples of global explanations are methods that use deconvolutional networks to characterize high-layer units in a CNN for image classification (Zeiler and Fergus 2014) and approaches that derive an identity for each filter in a CNN for text classification, in terms of the captured semantic classes (Jacovi, Sar Shalom, and Goldberg 2018).

Some Local Post-Hoc Explanation methods provide visual insights, for example through a GAN¹-generated image to assess the information detail of deep representations extracted from the input text (Spinks and Moens 2018), however, as these methods stemmed from efforts into making neural image classifiers more *readable*, they are usually designed to trace back the portions of the network input that mostly contributed to the output decision. Network propagation techniques are used to identify the patterns of a given input item (e.g., an image) that are linked to the particular deep neural network prediction as in (Erhan, Courville, and Bengio 2010; Zeiler and Fergus 2014). Usually, these are based on backward algorithms that layer-wise reuse arc weights to propagate the prediction from the output down to the input, thus leading to the recreation of *meaningful* patterns in the input space. Typical examples are deconvolution heatmaps, used to approximate through Taylor series the partial derivatives at each

1 A Generative Adversarial Network (GAN) (Goodfellow et al. 2014) is a class of machine learning systems in which two networks compete in a zero-sum game: the generator has to produce synthetic data, usually from a random input signal, while the discriminator has to detect if the input it is fed with comes from the real data or it has been produced from the generator.

layer (Simonyan, Vedaldi, and Zisserman 2014), or the so-called Layer-wise Relevance Propagation (LRP), that redistributes back positive and negative evidence across the layers (Bach et al. 2015).

Several efforts have been made in the perspective of providing explanations of a neural classifier, often by focusing into highlighting a handful of crucial features (Baehrens et al. 2010) or deriving simpler, more readable models from a complex one, e.g., a binary decision tree (Frosst and Hinton 2017), or by local approximation with linear models (Ribeiro, Singh, and Guestrin 2016). However, although they can explicitly show the representations learned in the specific hidden neurons (Frosst and Hinton 2017), these approaches base their effectiveness on the user ability to establish the quality of the reasoning and the accountability, as a side effect of the quality of the selected features: this can be very hard in tasks where no strong theory about the decision is available or the boundaries between classes are not well defined. Sometimes, explanations are associated to vector representations as in (Ribeiro, Singh, and Guestrin 2016), i.e., bag-of-words in case of text classification, which is clearly weak at capturing significant linguistic abstractions, such as the involved syntactic relations. When embeddings are used to trigger neural learning the readability of the model is a clear proof of the consistency of the adopted vectors as meaning representations, as clear understanding of what a numerical representation is describing allows human inspectors to assess whether the machine correctly modelled the target phenomena or not. Readability here refers to the property of a neural network to support *linguistically motivated explanations* about its (textual) inference. A recent methodology exploits the coupling of the classifier with some sort of generator, or decoder, responsible for the selection of output justifications: (Lei, Barzilay, and Jaakkola 2016) propose a generator that provides rationales for a multi-aspect sentiment analysis prediction by highlighting short and self-sufficient phrases in the original text.

Concerns in the research area of deriving interpretable, sparse representations from dense embeddings (Faruqui et al. 2015; Subramanian et al. 2018) have recently grown: for example, in (Trifonov et al. 2018) an effective unsupervised approach to disentangle meanings from embedding dimensions as well as automatic evaluation method have been proposed. In this work, we present a model generating *local post-hoc explanations* through analogies with previous real examples by exploiting the Layer-wise Relevance Propagation extended to a linguistically motivated neural architecture, the KDA, that exhibits a promising level of epistemological transparency. With respect to the works above, our proposal holds a few nice properties. First, the instance representation (i.e., the embedding) is derived from similarity scores estimated against real examples in the training set. Those depend on the general linguistic material as well as the task-relevant information (i.e., the target class), according to the underlying kernel. This enforces full adherence between the explanation-generation process and the decision-making process executed by the neural discriminator. Second, it is well suited to deal with short texts, where it may be difficult to highlight meaningful, yet not trivial, portions of input as justifications, as well as with the classification of segments of longer text (e.g., multi-aspect sentiment analysis) in a fashion similar to the one described for SRL in Section 6. Moreover, it provides explanations that are easily interpretable even by non-expert users, as they are inspired and expressed at language level: these are done by entire sentences and allow the human inspector to implicitly detect lexical, semantic and syntactic connections in the comparison, and consequently judge the trustworthiness of the decision, relying only on his/her linguistic competence. Lastly, the explanation-generation process is computationally inexpensive, as the LRP corresponds to a single

pass of backward propagation. As discussed in Section 5, it provides a transparent and epistemologically coherent view on the system's decision.

3. Kernel methods in semantic inferences

Prediction techniques such as Support Vector Machines learn decision surfaces that correspond to hyper-planes in the original feature space by computing inner products between input examples; consequently, they are inherently linear and cannot discover nonlinear patterns in data. A possible solution is to use a mapping $\phi : x \in \mathbb{R}^n \mapsto \phi(x) \in F \subseteq \mathbb{R}^N$ such that nonlinear relations in the original space become linearly separable in the target projection space, enabling the SVM to correctly separate the data by computing inner products $\langle \phi(x_i), \phi(x_j) \rangle$ in the new feature space. However, such projections can be computationally intense. *Kernel functions* are a class of functions that allow to compute $\langle \phi(x_i), \phi(x_j) \rangle$ without explicitly accessing the input representation in the projection space. Formally, given a feature space X and a mapping ϕ from X to F , a kernel κ is any function satisfying

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \quad \forall x_i, x_j \in X \quad (1)$$

An important generalization result is the Mercer Theorem (Shawe-Taylor and Cristianini 2004), stating that for any symmetric positive semi-definite function κ there exists a mapping ϕ such that 1 is satisfied. Hence, kernels include a broad class of functions (Shawe-Taylor and Cristianini 2004). Research community has been exploring kernel methods for decades and a wide variety of kernel paradigms have been proposed. In the following sub-sections, we will illustrate advancements in Tree-Kernels (TKs, (Collins and Duffy 2002b)), as they are well suited to encode formalisms, such as dependency graphs or grammatical trees, traditionally exploited in the linguistics communities.

3.1 Semantic Kernels

Learning to solve NLP tasks usually involves the acquisition of decision models based on complex semantic and syntactic phenomena. For instance, in Paraphrase Detection, verifying whether two sentences are valid paraphrases involves rewriting rules in which the syntax plays a fundamental role. In Question Answering, the syntactic information is crucial, as largely demonstrated in (Croce, Moschitti, and Basili 2011). Similar needs are applicable to the Semantic Role Labeling task, that consists in the automatic discovery of linguistic predicates (together with their corresponding arguments) in texts. A natural approach to such problems is to apply Kernel methods (Robert Müller et al. 2001; Shawe-Taylor and Cristianini 2004), that have been traditionally proposed to decouple similarity metrics and learning algorithms in order to alleviate the impact of feature engineering in inductive processes. Kernels may directly operate on complex structures and then be used in combination with linear learning algorithms, such as Support Vector Machines (SVM, (Vapnik 1998)). Sequences (Cancedda et al. 2003) or tree kernels (Collins and Duffy 2002a) are of particular interest as the feature space they capture reflects linguistic patterns. A sentence s can be represented as a parse tree that expresses the grammatical relations implied by s : parse trees are extracted by using the Stanford Parser (Manning et al. 2014). Tree kernels (TKs, (Collins and Duffy 2002a)) can be employed to directly operate on such parse trees, evaluating the tree fragments shared by the input trees. This operation corresponds to a dot product in the implicit feature space of all possible tree fragments. Whenever the dot product is available in

the implicit feature space, kernel-based learning algorithms, such as SVMs (Cortes and Vapnik 1995), can operate in order to automatically generate robust prediction models. TKs thus allow estimating the similarity among texts, directly from sentence syntactic structures, that can be represented by parse trees. The underlying idea is that the similarity between two trees T_1 and T_2 can be derived from the number of shared tree fragments. Let the set $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$ be the space of all the possible substructures and $\chi_i(n_2)$ be an indicator function that is equal to 1 if the target t_i is rooted at the node n_2 and 0 otherwise. A tree-kernel function over T_1 and T_2 is defined as follows: $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$ where N_{T_1} and N_{T_2} are the sets of nodes of T_1 and T_2 respectively, and $\Delta(n_1, n_2) = \sum_{k=1}^{|\mathcal{T}|} \chi_k(n_1) \chi_k(n_2)$ which computes the number of common fragments between trees rooted at nodes n_1 and n_2 . The feature space generated by the structural kernels obviously depends on the input structures. Notice that different tree representations embody different linguistic evidence and theories, and may produce more or less effective syntactic/semantic feature spaces for a given task.

Many available linguistic resources are enriched with formalisms dictated by Dependency grammars. They can produce a significantly different representation as exemplified in Figure 1. Since tree kernels are not tailored to model the labeled edges that are typical of dependency graphs, these latter are rewritten into explicit hierarchical representations. Different rewriting strategies are possible, as discussed in (Croce, Moschitti, and Basili 2011): a representation that is shown to be effective in several tasks is the Grammatical Relation Centered Tree (GRCT) illustrated in Figure 2: the PoS-Tags are children of grammatical function nodes and direct ancestors of their associated lexical items. Another possible representation is the Lexical Only Centered Tree (LOCT) shown in Figure 3, which contains only lexical nodes and the edges reflect some dependency relations.

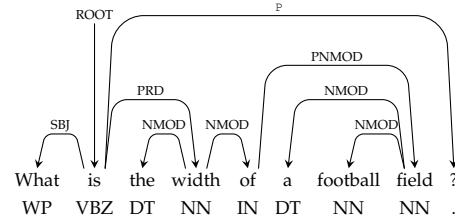


Figure 1
Dependency Parse Tree of "What is the width of a football field?".

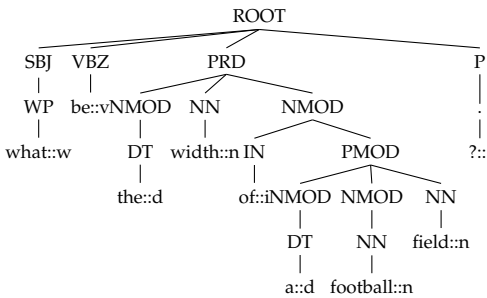


Figure 2
Grammatical Relation Centered Tree (GRCT) of "What is the width of a football field?".

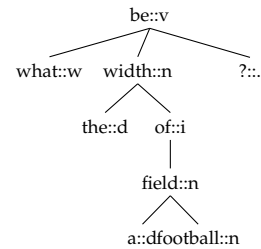


Figure 3
Lexical Only Centered Tree (LOCT) of "What is the width of a football field?".

Different tree kernels can be defined according to the types of tree fragments considered in the evaluation of the matching structures. Subset of trees are exploited by the *Subset Tree Kernel* (Collins and Duffy 2002a), which is usually referred to as Syntactic Tree Kernel (STK); they are more general structures since their leaves can be also non-terminal symbols. The subset trees satisfy the constraint that grammatical rules cannot be broken and every tree exhaustively represents a CFG rule. *Partial Tree Kernel* (PTK, (Moschitti 2006)) relaxes this constraint considering partial trees, i.e., fragments generated by the application of partial production rules (e.g. sequences of non terminal nodes with gaps). The strict constraint imposed by the STK may be problematic especially when the training dataset is small and only few syntactic tree configurations can be observed. Overcoming this limitation, the PTK usually leads to higher accuracy, as shown by (Moschitti 2006).

Capitalizing lexical semantic information in Convolution Kernels. The tree kernels introduced in previous section perform a hard match between nodes when comparing two substructures. In NLP tasks, when nodes are words, this strict requirement reflects in a too strict lexical constraint, that poorly reflects semantic phenomena, such as the synonymy of different words or the polysemy of a lexical entry. To overcome this limitation, we adopt Distributional models of Lexical Semantics (Sahlgren 2006; Schütze 1993; Padó and Lapata 2007) to generalize the meaning of individual words by replacing them with geometrical representations (also called Word Embeddings) that are automatically derived from the analysis of large-scale corpora (Mikolov et al. 2013). These representations are based on the idea that words occurring in the same contexts tend to have similar meaning: the adopted distributional models generate vectors that are spatially close when the associated words exhibit a similar usage in large-scale document collections. Under this perspective, the distance between vectors reflects semantic relations between the represented words, such as paradigmatic relations, e.g., quasi-synonymy.² These word spaces allow to define meaningful soft matching between lexical nodes, in terms of the distance between their representative vectors. As a result, it is possible to obtain more informative kernel functions, which are able to capture syntactic and semantic phenomena through grammatical and lexical constraints. Notice that the supervised setting of a learning algorithm (such as SVM), operating over the resulting kernel, is augmented with the word representations generated by the unsupervised distributional methods, thus characterizing a cost-effective semi-supervised paradigm.

The *Smoothed Partial Tree Kernel* (SPTK) described in (Croce, Moschitti, and Basili 2011) exploits this idea extending the PTK formulation with a similarity function σ between nodes:

$$\Delta_{SPTK}(n_1, n_2) = \mu\lambda\sigma(n_1, n_2), \text{ if } n_1 \text{ and } n_2 \text{ are leaves}$$

$$\Delta_{SPTK}(n_1, n_2) = \mu\sigma(n_1, n_2) \left(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2: l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{k=1}^{l(\vec{I}_1)} \Delta_{SPTK}(c_{n_1}(i_k^1), c_{n_2}(i_k^2)) \right) \quad (2)$$

In the SPTK formulation, the similarity function $\sigma(n_1, n_2)$ between two nodes n_1 and n_2 can be defined as follows:

- if n_1 and n_2 are both lexical nodes, then
 $\sigma(n_1, n_2) = \sigma_{LEX}(n_1, n_2) = \tau \frac{\vec{v}_{n_1} \cdot \vec{v}_{n_2}}{\|\vec{v}_{n_1}\| \|\vec{v}_{n_2}\|}$. It is the cosine similarity between

² In such spaces, vectors representing the nouns *football* and *soccer* will be near (as they are synonyms according to one of their senses) while *football* and *dog* are far

the word vectors \vec{v}_{n_1} and \vec{v}_{n_2} associated with the labels of n_1 and n_2 , respectively. τ is called *terminal factor* and weights the contribution of the lexical similarity to the overall kernel computation.

- else if n_1 and n_2 are nodes sharing the same label, then $\sigma(n_1, n_2) = 1$.
- else $\sigma(n_1, n_2) = 0$.

The decay factors λ and μ are responsible for penalizing large child subsequences (that can include gaps) and partial sub-trees that are deeper in the structure, respectively.

3.2 The Nyström approximation

Given an input training dataset \mathcal{D} of objects $o_i, i = 1 \dots N$, a kernel $K(o_i, o_j)$ is a similarity function over \mathcal{D}^2 that corresponds to a dot product in the implicit kernel space, i.e., $K(o_i, o_j) = \Phi(o_i) \cdot \Phi(o_j)$. The advantage of kernels is that the projection function $\Phi(o) = \vec{x} \in \mathbb{R}^n$ is never explicitly computed (Shawe-Taylor and Cristianini 2004). In fact, this operation may be prohibitive when the dimensionality n of the underlying kernel space is extremely large, as for Tree Kernels (Collins and Duffy 2002a). Kernel functions are used by learning algorithms, such as SVM, to operate only implicitly on instances in the kernel space, by never accessing their explicit definition. Let us apply the projection function Φ over all examples o_i from \mathcal{D} to derive representations, \vec{x}_i denoting the i -th row of the matrix \mathbf{X} . The Gram matrix can always be computed as $\mathbf{G} = \mathbf{X}\mathbf{X}^\top$, with each single element corresponding to $\mathbf{G}_{ij} = \Phi(o_i)\Phi(o_j) = K(o_i, o_j)$. The aim of the Nyström method (Drineas and Mahoney 2005) is to derive a new low-dimensional embedding $\tilde{\vec{x}}$ in a l -dimensional space, with $l \ll n$ so that $\tilde{\mathbf{G}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top$ and $\tilde{\mathbf{G}} \approx \mathbf{G}$. This is obtained by generating an approximation $\tilde{\mathbf{G}}$ of \mathbf{G} using a subset of l columns of the Gram matrix, i.e., the kernel evaluations between all the objects $\in \mathcal{D}$ and a selection of a subset $L \subset \mathcal{D}$ of the available examples, called *landmarks*. Suppose we randomly sample l columns of \mathbf{G} , and let $\mathbf{C} \in \mathbb{R}^{N \times l}$ be the matrix of these sampled columns. Then, we can rearrange the columns and rows of \mathbf{G} and define $\mathbf{X} = [\mathbf{X}_1 \ \mathbf{X}_2]$ such that:

$$\mathbf{G} = \mathbf{X}\mathbf{X}^\top = \begin{bmatrix} \mathbf{W} & \mathbf{X}_1^\top \mathbf{X}_2 \\ \mathbf{X}_2^\top \mathbf{X}_1 & \mathbf{X}_2^\top \mathbf{X}_2 \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{X}_2^\top \mathbf{X}_1 \end{bmatrix}$$

where \mathbf{X}_1 includes rows for the subset of \mathbf{G} that contains only landmarks, $\mathbf{W} = \mathbf{X}_1^\top \mathbf{X}_1$ is their corresponding similarity matrix and finally \mathbf{C} kernel evaluations between landmarks and the remaining examples. The Nyström approximation can be defined as:

$$\mathbf{G} \approx \tilde{\mathbf{G}} = \mathbf{C}\mathbf{W}^\dagger \mathbf{C}^\top \quad (3)$$

where \mathbf{W}^\dagger denotes the Moore-Penrose inverse of \mathbf{W} . The Singular Value Decomposition (SVD) is used to obtain \mathbf{W}^\dagger as follows. First, \mathbf{W} is decomposed so that $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, where \mathbf{U} and \mathbf{V} are both orthogonal matrices, and \mathbf{S} is a diagonal matrix containing the (non-zero) singular values of \mathbf{W} on its diagonal. Since \mathbf{W} is symmetric and positive definite, it holds that $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$. Then, $\mathbf{W}^\dagger = \mathbf{U}\mathbf{S}^{-1}\mathbf{U}^\top = \mathbf{U}\mathbf{S}^{-\frac{1}{2}}\mathbf{S}^{-\frac{1}{2}}\mathbf{U}^\top$ and the Equation 3 can be rewritten as

$$\mathbf{G} \approx \tilde{\mathbf{G}} = \mathbf{C}\mathbf{U}\mathbf{S}^{-\frac{1}{2}}\mathbf{S}^{-\frac{1}{2}}\mathbf{U}^\top \mathbf{C}^\top = (\mathbf{C}\mathbf{U}\mathbf{S}^{-\frac{1}{2}})(\mathbf{C}\mathbf{U}\mathbf{S}^{-\frac{1}{2}})^\top = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top \quad (4)$$

which explicitates the desired approximation of G in terms of the described decomposition. Given an input example $o \in \mathcal{D}$, a new low-dimensional representation \tilde{x} can be thus determined by considering the corresponding item of \mathbf{C} as

$$\tilde{x} = \vec{c} \mathbf{U} \mathbf{S}^{-\frac{1}{2}} \quad (5)$$

where \vec{c} is the vector whose j -th individual component contains the evaluation of the kernel function between o and the landmark $o_j \in L$. Therefore, the method produces l -dimensional vectors.

As discussed in the next section, the Nyström method is a crucial step in our approach, as the resulting representation is inherently connected with the task at hand (each dimension of the input is linked with a real example, hence with a target class) and depends on the shared properties, between the original input example and the landmarks, captured by the exploited kernel. Notice that, while we investigated only tree linguistic kernels (as they are a natural choice in NLP), our approach can be in principle extended with any suitable kernel, as long as the captured similarities produce a satisfying representation (effectiveness of other kernel functions will be investigated in future works).

4. Interpretable Kernel-Based Deep Architectures

4.1 Kernel-based Deep Architectures

As discussed in Section 3.2, the Nyström representation \tilde{x} of any input example o is linear and can be adopted to feed a neural network architecture. We assume a labeled dataset $\mathcal{L} = \{(o, y) \mid o \in \mathcal{D}, y \in Y\}$ is available, where o refers to a generic instance and y is its associated class. In this Section, we define a Multi-Layer Perceptron (MLP) architecture, with a specific Nyström layer based on the Nyström embeddings of Eq. 5. We will refer to this architecture, shown in Figure 4, as Kernel-based Deep Architecture (KDA). KDA has an *input layer*, a *Nyström layer*, a possibly empty sequence of non-linear *hidden layers* and a final *classification layer*, which produces the output.

The *input layer* corresponds to the input vector \vec{c} , i.e., the row of the \mathbf{C} matrix associated to an example o . Notice that, for adopting the KDA, the values of the matrix \mathbf{C} should be all available. In the training stage, these values are in general cached. During the classification stage, the \vec{c} vector corresponding to an example o is directly computed by l kernel computations between o and each of the l landmarks.

The input layer is mapped to the *Nyström layer*, through the projection in Equation 5. Notice that the embedding provides also the proper weights, defined by $\mathbf{U} \mathbf{S}^{-\frac{1}{2}}$, so that the mapping can be expressed through the Nyström matrix $\mathbf{H}_{Ny} = \mathbf{U} \mathbf{S}^{-\frac{1}{2}}$: it corresponds to a pre-trained stage derived through SVD, as discussed in Section 3.2. Equation 5 provides a static definition for \mathbf{H}_{Ny} whose weights can be left invariant during the neural network training. However, the values of \mathbf{H}_{Ny} can be made available for the standard back-propagation adjustments applied for training. Formally, the low-dimensional embedding of an input example o , is $\tilde{x} = \vec{c} \mathbf{H}_{Ny} = \vec{c} \mathbf{U} \mathbf{S}^{-\frac{1}{2}}$.

The resulting outcome \tilde{x} is the input to one or more non-linear *hidden layers*. Each t -th hidden layer is realized through a matrix $\mathbf{H}_t \in \mathbb{R}^{h_{t-1} \times h_t}$ and a bias vector $\vec{b}_t \in \mathbb{R}^{1 \times h_t}$, whereas h_t denotes the desired hidden layer dimensionality. Clearly, given that $\mathbf{H}_{Ny} \in \mathbb{R}^{l \times l}$, $h_0 = l$. The first hidden layer in fact receives in input $\tilde{x} = \vec{c} \mathbf{H}_{Ny}$, that corresponds to $t = 0$ layer input $\tilde{x}_0 = \tilde{x}$ and its computation is formally expressed by

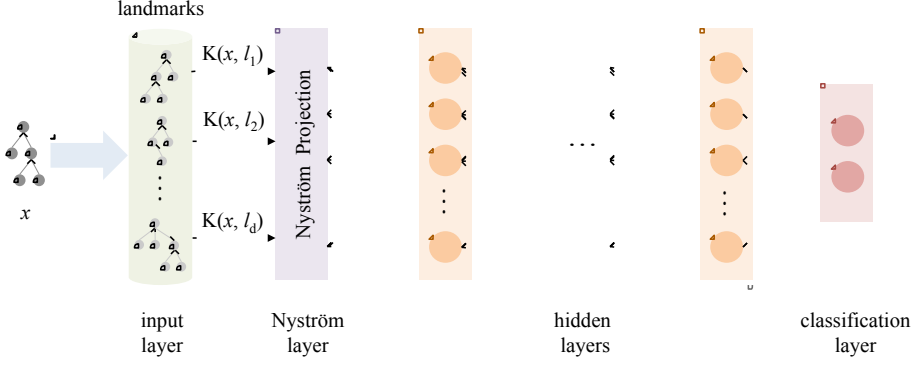


Figure 4
Kernel-based Deep Architecture.

$\vec{x}_1 = f(\vec{x}_0 \mathbf{H}_1 + \vec{b}_1)$, where f is a non-linear activation function, here a Rectified Linear Unit (ReLU). In general, the generic t -th layer is modeled as:

$$\vec{x}_t = f(\vec{x}_{t-1} \mathbf{H}_t + \vec{b}_t) \quad (6)$$

The final layer of KDA is the *classification layer*, realized through the output matrix \mathbf{H}_O and the output bias vector \vec{b}_O . Their dimensionality depends on the dimensionality of the last hidden layer (called O_{-1}) and the number $|Y|$ of different classes, i.e., $\mathbf{H}_O \in \mathbb{R}^{h_{O-1} \times |Y|}$ and $\vec{b}_O \in \mathbb{R}^{1 \times |Y|}$, respectively. In particular, this layer computes a linear classification function with a softmax operator so that $\hat{y} = \text{softmax}(\vec{x}_{O-1} \mathbf{H}_O + \vec{b}_O)$.

In order to avoid over-fitting, two different regularization schemes are applied. First, the dropout is applied to the input \vec{x}_t of each hidden layer ($t \geq 1$) and to the input \vec{x}_{O-1} of the final classifier. Second, a L_2 regularization is applied to the norm of each layer.

Finally, the KDA is trained by optimizing a loss function made of the sum of two factors: first, the cross-entropy function between the gold classes and the predicted ones; second the L_2 regularization, whose importance is regulated by a meta-parameter λ . The final loss function is thus

$$L(y, \hat{y}) = \sum_{(o, y) \in \mathcal{L}} y \log(\hat{y}) + \lambda \sum_{\mathbf{H} \in \{\mathbf{H}_t\} \cup \{\mathbf{H}_O\}} \|\mathbf{H}\|^2$$

where \hat{y} are the softmax values computed by the network and y are the true one-hot encoding values associated with the example from the labeled training dataset \mathcal{L} .

4.2 Layer-wise Relevance Propagation

Layer-wise Relevance propagation (LRP, presented in (Bach et al. 2015)) is a framework which allows to decompose the prediction of a deep neural network computed over a sample, e.g. an image, down to relevance scores for the individual input dimensions of the sample such as subpixels of an image.

More formally, let $f : \mathbb{R}^d \rightarrow \mathbb{R}^+$ be a positive real-valued function taking a vector $x \in \mathbb{R}^d$ as input. The function f can quantify, for example, the probability of x being

in a certain class. The Layer-wise Relevance Propagation assigns to each dimension, or feature, x_d a relevance score $R_d^{(1)}$ such that:

$$f(x) \approx \sum_d R_d^{(1)} \quad (7)$$

Features whose score is $R_d^{(1)} > 0$ or $R_d^{(1)} < 0$ correspond to evidence in favor or against, respectively, the output classification. In other words, LRP allows to identify fragments of the input playing key roles in the decision, by propagating relevance backwards. Let us suppose to know the relevance score $R_j^{(l+1)}$ of a neuron j at network layer $l + 1$, then it can be decomposed into messages $R_{i \leftarrow j}^{(l,l+1)}$ sent to neurons i in layer l :

$$R_j^{(l+1)} = \sum_{i \in (l)} R_{i \leftarrow j}^{(l,l+1)} \quad (8)$$

Hence it derives that the relevance of a neuron i at layer l can be defined as:

$$R_i^{(l)} = \sum_{j \in (l+1)} R_{i \leftarrow j}^{(l,l+1)} \quad (9)$$

Note that 8 and 9 are such that 7 holds. In this work, we adopted the ϵ -rule defined in (Bach et al. 2015) to compute the messages $R_{i \leftarrow j}^{(l,l+1)}$:

$$R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j + \epsilon \cdot \text{sign}(z_j)} R_j^{(l+1)}$$

where $z_{ij} = x_i w_{ij}$ and $\epsilon > 0$ is a numerical stabilizing term and must be small. The informative value is justified by the fact that the weights w_{ij} are linked to the weighted activations of the input neurons.

If we apply the above process to the KDA applied to a linguistic inference task, e.g. sentence classification, then LRP implicitly traces back the syntactic, semantic and lexical relations between the decision and the landmarks: it thus selects the landmarks that were the most influential for the predicted structure, e.g. for deciding the class of the underlying sentence. Indeed, each landmark is uniquely associated to an entry of the input vector \vec{c} , as illustrated in Sec 4.1.

5. Generating explanations for predictions of deep models

Justifications for the KDA decisions can be obtained by explaining the evidence in favor or against a class using the set \mathcal{L} of landmarks as examples. The idea is to select those $l \in \mathcal{L}$ that the LRP method detects as the most active elements in layer 0 during the classification. Once such active landmarks are detected, an *Explanatory Model* is a function in charge of compiling a linguistically fluent explanation by using analogies (or differences) with the input case. The semantic expressiveness of such analogies makes the resulting explanation clear and increases the user confidence on the system reliability. When a sentence s is classified, LRP assigns activation scores r_ℓ^s to each individual landmark ℓ : let $\mathcal{L}^{(+)}$ (or $\mathcal{L}^{(-)}$) denote the set of landmarks with positive (or negative) activation score.

Formally, every explanation is characterized by a triple $e = \langle s, C, \tau \rangle$ where s is the input sentence, C is the predicted label and τ is the modality of the explanation: $\tau = +1$ for positive (i.e. acceptance) statements while $\tau = -1$ correspond to rejections of the decision C . A landmark ℓ is *positively activated* for a given sentence s if there are not more than $k - 1$ other active landmarks ℓ' whose activation value is higher than the one for ℓ , i.e.

$$|\{\ell' \in \mathcal{L}^{(+)} : \ell' \neq \ell \wedge r_{\ell'}^s \geq r_{\ell}^s > 0\}| < k$$

Similarly, a landmark ℓ is *negatively activated* when:

$$|\{\ell' \in \mathcal{L}^{(-)} : \ell' \neq \ell \wedge r_{\ell'}^s \leq r_{\ell}^s < 0\}| < k$$

k is a parameter used to make explanation depend on not more than k landmarks, denoted by \mathcal{L}_k . Positively (or negative) active landmarks in \mathcal{L}_k are assigned to an activation value $a(\ell, s) = +1$ (-1). $a(\ell, s) = 0$ for all other not activated landmarks.

Given the explanation $e = \langle s, C, \tau \rangle$, a landmark ℓ whose (known) class is C_{ℓ} is *consistent* (or *inconsistent*) with e according to the fact that the following function:

$$\delta(C_{\ell}, C) \cdot a(\ell, q) \cdot \tau$$

is positive (or negative, respectively), where $\delta(C', C) = 2\delta_{kron}(C' = C) - 1$ and δ_{kron} is the Kronecker delta. The *explanatory model* is then a function $M(e, \mathcal{L}_k)$ which maps an explanation e , a sub set \mathcal{L}_k of the active *and* consistent landmarks \mathcal{L} for e into a sentence f in natural language. Note that the value of k determines the amount of consistent landmarks and hence it regulates the tradeoff between the capacity of the system to produce an explanation at all and the adherence of such explanation to the machine inference process: low values of k grant that the Model generates explanations using landmarks with high activation scores only, however they may also result in the Model being unable to produce any explanation for some decisions, i.e., when no consistent landmark is available.

Of course several definitions for $M(e, \mathcal{L}_k)$ are possible. A general explanatory model would be:

$$M(e, \mathcal{L}_k) = M(\langle s, C, \tau \rangle, \mathcal{L}_k) = \begin{cases} "s \text{ is } C \text{ since it recalls me of } \ell" \\ \forall \ell \in \mathcal{L}_k^+ \quad \text{if } \tau > 0 \\ \\ "s \text{ is not } C \text{ since it does not recall me of} \\ \ell \text{ which is } C" \\ \forall \ell \in \mathcal{L}_k^- \quad \text{if } \tau < 0 \\ \\ "s \text{ is } C \text{ but I don't know why}" \\ \text{if } \mathcal{L} \equiv \emptyset \end{cases}$$

where \mathcal{L}_k^+ and \mathcal{L}_k^- are the partition of landmarks with positive and negative relevance scores in \mathcal{L}_k , respectively.

Here we defined 3 explanatory models we used during experimental evaluation:

(*Basic Model*). The first model is the simplest. It returns an analogy only with the (unique) consistent landmark with the highest positive score if $\tau = 1$ and lowest negative when

$\tau = -1$. In case no active and consistent landmark can be found, the Basic model returns a phrase stating only the predicted class, with no explanation. As an example, the explanation of an accepted decision in an Argument Classification task, described by the triple $e_1 = \langle \text{'Put this plate in the center of the table'}, \text{THEME}_{\text{PLACING}}, 1 \rangle$, the model would produce:

Example

*I think “this plate” is THEME of PLACING in “Robot put **this plate** in the center of the table” since it reminds me of “the soap” in “Can you put **the soap** in the washing machine?”.*

(*Multiplicative Model*). In a second model, denoted as *multiplicative*, the system makes reference to up to $k_1 \leq k$ analogies with positively active and consistent landmarks. Given the above explanation e_1 , and $k_1 = 2$, it would return:

Example

*I think “this plate” is THEME of PLACING in “Robot put **this plate** in the center of the table” since it reminds me of “the soap” in “Can you put **the soap** in the washing machine?” and it also reminds me of “my coat” in “hang **my coat** in the closet in the bedroom”.*

(*Contrastive Model*). The last model we propose is more complex, since it returns both a positive (where $\tau = 1$) and a negative ($\tau = -1$) analogy by selecting, respectively, the most positively relevant and the most negatively relevant consistent landmark. For instance it could result in the following explanation:

Example

*I think “this plate” is the THEME of PLACING in “Robot put **this plate** in the center of the table” since it reminds me of “the soap” which is in “Can you put **the soap** in the washing machine” and it is not the GOAL of PLACING since different from “on the counter” in “put the plate **on the counter**”.*

All the three models find their foundations, from argumentation theory, in a “argument by analogy” schema (Walton, Reed, and Macagno 2008). Such kind of arguments gains strength proportionally to the linguistic plausibility of the analogy: the user exposed is thus expected to implicitly gauge evidences from the linguistic properties shared between the input sentence (or its parts) and the one used for comparison. Moreover, the higher their importance (e.g. strength of activation) with respect to the output decision the larger the amount of trust in the machine verdict, accordingly.

5.1 Evaluating the quality of an explanation

In general, judging the semantic coherence of an explanation is a very difficult task. In this section we propose an approach which aims at evaluating the quality of the explanations in terms of the amount of information that a user would gather given an explanation with respect to a scenario where such explanation is not made available. More formally, let $P(C|s)$ and $P(C|s, e)$ be, respectively, the prior probability of the user believing that the classification of s is correct and the probability of the user believing that the classification of s is correct given an explanation. Note that both indicate the level of confidence the user has in the classifier (i.e. the KDA) given the amount of

available information, i.e. with and without explanation. Three kinds of explanations are possible:

- **Useful explanations:** these are explanations such that C is correct and $P(C|s, e) > P(C|s)$ or C is not correct and $P(C|s, e) < P(C|s)$
- **Useless explanations:** they are explanations such that $P(C|s, e) = P(C|s)$
- **Misleading explanations:** they are explanations such that C is correct and $P(C|s, e) < P(C|s)$ or C is not correct and $P(C|s, e) > P(C|s)$

The core idea is that semantically coherent and exhaustive explanations must indicate correct classifications whereas incoherent or non-existent explanations must hint towards wrong classifications. Given the above probabilities, we can measure the quality of an explanation by computing the *Information Gain* (Kononenko and Bratko 1991) achieved: the *posterior* probability is expected to grow w.r.t. to the *prior* one for correct decisions when a good explanation is available against the input sentence, while decreasing for bad or confusing explanations. The intuition behind Information Gain is that it measures the amount of information (provided in number of bits) gained by the explanation about the decision of accepting the system decision about an incoming sentence s . A positive gain indicates that the probability amplifies towards the right decisions, and declines with errors. We will let users to judge the quality of the explanation and assign them a posterior probability that increases along with better judgments. In this way we have a measure of how convincing the system is about its decisions as well as how weak it is in clarifying erroneous cases. To compare the overall performance of the different explanatory models M , the Information Gain is measured against a collection of explanations generated by M and then normalized throughout the collection's entropy E as follows:

$$I_r = \frac{1}{E} \frac{1}{|\mathcal{T}_s|} \sum_{j=1}^{|\mathcal{T}_s|} I(j) = \frac{I_a}{E} \quad (10)$$

where \mathcal{T}_s is the explanations collection and $I(j)$ is the Information Gain of explanation j .

6. Experimental Investigations

The effectiveness of the proposed approach has been measured against the Argument Classification task in the Semantic Role Labeling chain (SRL, (Palmer, Gildea, and Xue 2010)), consisting in the detection of the semantic arguments associated with the predicate of a sentence and their classification into their specific roles (Fillmore 1985). For example, given the sentence "*Bring the fruit onto the dining table*", the task would be to recognize the verb "*bring*" as evoking the BRINGING frame, with its roles, THEME for "*the fruit*" and GOAL for "*onto the dining table*". Argument classification corresponds to the subtask of assigning labels to the sentence fragments spanning individual roles. In particular, we tested our system against two datasets, consisting of domotic commands (i.e. HuRIC, (Bastianelli et al. 2014, 2016)), in English and Italian respectively.

To evaluate the performances of proposed explanation models, we associated values for the posteriori probability $P(C|s, e)$ to five defined labels: *Very Good* if the provided explanation is clearly convincing, *Good* if the explanation is convincing but it is

Table 1

Posterior probabilities w.r.t. quality categories.

Category	$P(C s, e)$	$1 - P(C s, e)$
V.Good	0.95	0.05
Good	0.8	0.2
Weak	0.5	0.5
Bad	0.2	0.8
Incoher.	0.05	0.95

Table 2

Weights for the Cohen's Kappa κ_w statistics.

Class	Incoher.	Bad	Weak	Good	V.Good
Incoher.	1.00	0.83	0.50	0.16	0.00
Bad	0.83	1.00	0.66	0.33	0.16
Weak	0.50	0.66	1.00	0.66	0.50
Good	0.16	0.33	0.66	1.00	0.83
V.Good	0.00	0.16	0.50	0.83	1.00

Table 3

Information gains for the three Explanatory Models applied to the SRL-AC datasets in English and Italian, respectively.

	Basic	Multiplicative	Contrastive	<i>accuracy</i>
SRL-AC eng	0.669	0.663	0.667	0.961
SRL-AC ita	0.561	0.632	0.651	0.912

not completely related to the input example so that some doubts about the system decision still remain, *Uncertain* if the explanation is not useful to increase the confidence of the user with respect to the system decision, *Bad* if the explanation makes the annotator believe that the system decision is not correct while *Incoherent* corresponds to the case where the explanation is clearly inconsistent with the input example and suggests a clear error of the system in providing its answer. Corresponding values are shown in Table 1.

We gathered into explanation datasets hundreds of explanations from the three models for each task and asked annotators to perform independent labeling of them, with external knowledge only about the overall balance between explanations from incorrect and correct predictions. In case of multiple annotators, we addressed their consensus by measuring a weighted Cohen's Kappa (adopting the weights reported in Table 2).

6.1 Argument Classification in English

For the English language, the dataset included over 650 annotated transcriptions of spoken robotic commands, organized in 18 frames and about 60 arguments. We extracted single arguments from each HuRIC example, for a total of 1,300 instances. We performed extensive 10-fold cross-validation, optimizing network hyper-parameters

via grid-search for each test set. Consistently with (Croce, Moschitti, and Basili 2011), we generated Nyström representation from 200 landmarks exploiting an equally-weighted linear combination of Smoothed Partial Tree Kernel function, operating over Grammatical Relation Centered Tree (GRCT) derived from dependency grammar, with default parameters $\mu = \lambda = 0.4$, and a linear kernel function applied to sparse vector representing the instance's frame. With these settings, the KDA accuracy was 96.1%. Among the available examples, we sampled 692 explanations equally balanced among true positives, false positives, false negatives and true negatives. Due to the required balanced representation of all classes, the prior probability of the sample thus corresponds to an entropy ~ 1 . Two annotators (measured Cohen's kappa is 0.783) were exposed to partially overlapping selections from the overall collection of explanations, such that explanations from the 3 models were equally distributed between the two, in order to mitigate human biases. Results are shown in Table 3. In this task, all models were able to gain more than two thirds of needed information. The alike scores of the three models are probably due to the narrow linguistic domain of the corpus and the well-defined semantic boundaries between the arguments.

In a scenario such as domestic Human Robotic Interfaces, the quality of individual explanatory models is very important as the robot is made capable of using explanation in a dialogue with the user. Let us consider the following examples obtained by the contrastive model:

Example

I think “the washer” is the CONTAINING OBJECT of CLOSURE in “Robot can you open the washer?” since it reminds me of “the jar” in “close the jar” and it is not the THEME of BRINGING since different from “the jar” in “take the jar to the table of the kitchen”.

This argumentation is very rich. It must be observed that it is not just the result of the text similarity between the examples and the question, something that is usually directly expressed by the kernel. In the example, the lexical overlap between the command and the explanation is very limited. Rather, the explanation is strictly dependent on the model and on the instance. The command cited is the one activated by the feedforward process in the KDA, i.e. the one that has been found useful in the inference. This is a dynamic side effect of the KDA model and has a dynamic nature that changes across different cases. In the situation

Example

I think “me” is the BENEFICIARY of BRINGING in “I would like some cutlery can you get me some?” since reminds me of “me” in “bring me a fork from the press.” and it is not the COTHEME of COTHEME since different from “me” in “Would you please follow me to the kitchen?”.

the role of grammatical information is more explicit also in the counterargument regarding the sentence *Would you please follow me to the kitchen?*

Both the above commands have limited lexical overlap with the retrieved landmarks. Nevertheless, the retrieved analogies make the explanations quite effective: an explanatory model such as the contrastive one seems to successfully capture semantic and syntactic relations among input instances and closely related landmarks that are meaningful and epistemologically clear.

6.2 Argument Classification in Italian

Evaluation also targeted a second dataset, that is the Italian HuRIC dataset (Bastianelli et al. 2016), including about 240 domotic commands, comprising of about 450 roles. As GRCT representations of instances were not available, we designed a tree representation reflecting the semantic frame structure, i.e. each sentence span corresponds to a non-lexical node labeled either as *lexical unit*, *argument* (assigned to the argument to be classified) or *other* (assigned to all other arguments). The Nyström projection was performed from a sampling of 100 landmarks. Again, this was combined with a linear kernel on sparse vector representations reflecting the instance's frame. The measured accuracy is 91.2% on about 90 examples, while the training and development set have a size of, respectively, 270 and 90 examples. Due to the small size of the data set and the high accuracy, we choose to generate an explanation dataset with a uncorrect-to-correct ratio of about 0.3, that is 144 explanations from correct predictions and 48 explanations from wrong predictions. A single annotator performed the manual labeling task, whose results are shown in Table 3: again, the information gain scores suggest that the generated explanations were able to correctly assist the human inspector in trusting or not the network decision. The slightly lower performances may be due to the skewness of the dataset, which penalizes good explanations from correct predictions, being the prior probability higher (in this case, $P(C, s) = 0.75$).

Nevertheless, the produced explanations for decisions over Italian sentences exhibit similar properties to their English counterpart. For example, consider

Example

I think “lo” is the PHENOMENON of LOCATING in “Ho bisogno del mio orologio cerca lo per favore” since reminds me of “una maglietta” in “Puoi cercare una maglietta rossa nell’armadietto”

Except for the lexical unit, there is little lexical overlapping between the two sentences, which also have a quite different syntax. The system is also able to distinguish between different roles sharing the same lexical surface, as in:

Example

I think “dal tavolo” is not the SOURCE of BRINGING in “Vai nella sala da pranzo prendi tutti i piatti dal tavolo e mettili nella lavastoviglie” since does not remind me of “dal tavolo” in “Porta mi il mio libro dal tavolo” but rather it is the SOURCE of TAKING since it reminds me of “dall’appendiabiti” in “Prendi il mio giacchetto dall’appendiabiti nella mia camera da letto”.

7. Conclusions

This paper extends the approach proposed in (Croce, Rossini, and Basili 2019), investigating its effectiveness in Semantic Role Labeling both for English and Italian. The proposed methodology exploits (Nyström) approximations for semantic kernel to derive vector embeddings able to support the explanation of quantitative linguistic inferences, such as those provided by neural networks. Produce explanations correspond to argumentations in natural language using analogy schemas with activated real training examples, i.e., landmarks. In order to assess the quality of compiled justification, an evaluation methodology based on Information Theory is applied. In particular, performances are measured with respect to increase in the information gain, that is decrease in entropy. Experimental results show how explanatory models provide helpful contribution to the confidence of the user in the network decision for

both languages: the explanations augment confidence in correct decisions and lower down the confidence for the network errors. Given that KDA and in particular the proposed Nyström embeddings can be largely used for epistemologically clear neural learning in natural language processing, we think that they correspond to meaningful embeddings with huge potential for better neural learning models. First, they promote language semantics in a natural way and create associations between input instances and decisions that are harmonic with respect human (logical) intuition. In a sense, linguistic inferences are explained without necessarily moving out of the language level. Second, they are mathematically solid models for different levels of language semantics according to different kernel formulations. In this way the embeddings can be fine tuned to tasks, without impacting on the learning architecture but only by modeling different aspects of language syntax and semantics in the kernel function. Finally, the explanations proposed in this paper correspond just to an early stage of the research. In fact, there are many ways in which activated landmarks can be made useful in the explanation process and we are in a very early stage of such an exploration. For example, argumentation theory, as applied to the landmarks active in a decision and the source input example, can provide very rich ways to compile justification, i.e. short texts that argue for a decision.

References

- Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, Wojciech Samek, and Óscar Déniz Suárez. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. In *PloS one*, volume 10.
- Baehrens, David, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11.
- Bastianelli, Emanuele, Giuseppe Castellucci, Danilo Croce, Luca Iocchi, Roberto Basili, and Daniele Nardi. 2014. Huric: a human robot interaction corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May. European Language Resources Association (ELRA).
- Bastianelli, Emanuele, Danilo Croce, Andrea Vanzo, Roberto Basili, and Daniele Nardi. 2016. A discriminative approach to grounded spoken language understanding in interactive robotics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2747–2753, New York, New York, USA, July.
- Bengio, Yoshua, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August.
- Cancedda, Nicola, Éric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.
- Collins, Michael and Nigel Duffy. 2002a. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 625–632.
- Collins, Michael and Nigel Duffy. 2002b. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.
- Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12.
- Cortes, Corinna and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3).
- Croce, Danilo, Simone Filice, Giuseppe Castellucci, and Roberto Basili. 2017. Deep learning in semantic kernel spaces. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, July. Association for Computational Linguistics.
- Croce, Danilo, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 Conference on Empirical*

- Methods in Natural Language Processing*, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Croce, Danilo, Daniele Rossini, and Roberto Basili. 2019. Neural embeddings: accurate and readable inferences based on semantic kernels. *Natural Language Engineering*, 25(4):519–541.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Drineas, Petros and Michael W. Mahoney. 2005. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6.
- Erhan, Dumitru, Aaron Courville, and Yoshua Bengio. 2010. Understanding representations learned in deep architectures. Technical Report 1355, Université de Montréal/DIRO.
- Faruqui, Manaal, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, Beijing, China, July. Association for Computational Linguistics.
- Fillmore, Charles J. 1985. Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2):222–254.
- Frosst, Nicholas and Geoffrey E. Hinton. 2017. Distilling a neural network into a soft decision tree. In Tarek R. Besold and Oliver Kutz, editors, *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017)*, Bari, Italy, November 16th and 17th, 2017, volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Goldberg, Yoav. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57.
- Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., pages 2672–2680.
- Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8).
- Jacovi, Alon, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium, November. Association for Computational Linguistics.
- Kim, Yoon. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October. Association for Computational Linguistics.
- Kononenko, Igor and Ivan Bratko. 1991. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6(1).
- Lei, Tao, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November. Association for Computational Linguistics.
- Li, Xin and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3).
- Lipton, Zachary C. 2018. The mythos of model interpretability. *Queue*, 16(3), June.
- Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Moschitti, Alessandro. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*, Berlin, Germany, September.
- Moschitti, Alessandro. 2012. State-of-the-art kernels for natural language processing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Tutorial*

- Abstracts*, page 2, Jeju Island, Korea, July. Association for Computational Linguistics.
- Padó, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2), June.
- Palmer, M.S., D. Gildea, and N. Xue. 2010. *Semantic Role Labeling*. IEEE Morgan & Claypool Synthesis eBooks Library. Morgan & Claypool Publishers.
- Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.
- Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pages 1135–1144, San Francisco, California, USA, August, 13-17. ACM.
- Robert Müller, Klaus, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.
- Sahlgren, Magnus. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.
- Schütze, Hinrich. 1993. Word space. In *Advances in Neural Information Processing Systems 5*. Morgan-Kaufmann.
- Shawe-Taylor, John and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.
- Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Workshop Track Proceedings*, Banff, AB, Canada, April 14-16.
- Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Spinks, Graham and Marie-Francine Moens. 2018. Evaluating textual representations through image generation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 30–39, Brussels, Belgium, November. Association for Computational Linguistics.
- Strubell, Emma, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium, October-November.
- Subramanian, Anant, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *Proceedings of the Thirty Second AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, Louisiana USA, February.
- Trifonov, Valentin, Octavian-Eugen Ganea, Anna Potapenko, and Thomas Hofmann. 2018. Learning and evaluating sparse interpretable sentence embeddings. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 200–210, Brussels, Belgium, November. Association for Computational Linguistics.
- Vapnik, Vladimir N. 1998. *Statistical Learning Theory*. Wiley-Interscience.
- Walton, Douglas, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.
- Williams, Christopher K. I. and Matthias Seeger. 2001. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13 (NIPS 2000)*. MIT Press, pages 682–688.
- Zeiler, Matthew D. and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *13th European Conference on Computer Vision, ECCV 2014*, pages 818–833, Zurich, Switzerland, September.

State-of-the-art Italian dependency parsers based on neural and ensemble systems

Oronzo Antonelli*
Università di Bologna

Fabio Tamburini**
Università di Bologna

In this paper we present a work which aims to test the most advanced, state-of-the-art syntactic dependency parsers based on deep neural networks (DNN) on Italian. We made a large set of experiments by using two Italian treebanks containing different text types downloaded from the Universal Dependencies project and propose a new solution based on ensemble systems. We implemented the proposed ensemble solutions by testing different techniques described in literature, obtaining very good parsing results, well above the state of the art for Italian.

1. Introduction

Syntactic parsing of morphologically rich languages like Italian often poses a number of hard challenges. Various works applied different kinds of freely available parsers on Italian training them using different treebank resources and different methods to compare their results (Lavelli 2014; Alicante et al. 2015; Lavelli 2016) and gather a clear picture of the syntactic parsing task performance for the Italian language. In this direction, it seems relevant to cite the EVALITA¹ periodic campaigns for the evaluation of constituency and dependency parsers devoted to the syntactic analysis of Italian (Bosco and Mazzei 2011; Bosco et al. 2014).

Other studies regarding the syntactic parsing of English (Nivre and McDonald 2008; Surdeanu and Manning 2010) or Italian (Lavelli 2013; Mazzei 2015) tried to enhance the parsing performance by building some kind of *ensemble systems*.

By looking at the cited papers we can observe that they evaluated the state-of-the-art parsers before the “neural networks revolution” not including, with few exceptions, the last improvements proposed by new research studies.

The goal of this paper is twofold: first we would like to test the effectiveness of parsers based on the newly-proposed technologies, mainly deep neural networks, on Italian; second, we would like to propose an ensemble system able to further improve the neural parsers performance when parsing Italian texts.

This paper is structured as follows: Section 2 describes the architectures of the nine parsers we tested; Section 3 illustrates the datasets we employed for the evaluations; in Section 4 we will show the results of the single parsers evaluation while in Section 5 we will describe the different kind of ensemble systems that we propose to evaluate using the same datasets. In Section 6 we will draw some provisional conclusions.

* Dept. of Computer Science and Engineering - Mura Anteo Zamboni 7, 40126 Bologna, Italy.
E-mail: antonelli.aronzo@gmail.com

** Dept. of Classic Philology and Italian Studies - Via Zamboni 32, 40126 Bologna, Italy.
E-mail: fabio.tamburini@unibo.it

¹ <http://www.evalita.it>

2. The Neural Dependency Parsers

We considered nine state-of-the-art parsers representing a wide range of contemporary approaches to dependency parsing whose architectures are based on neural network models.

In (Chen and Manning 2014) it is proposed, for the first time, to represent words, Part-of-speech (PoS) tags and dependency types through a dense encoding using embedding vectors. In this way it is possible to automate the learning process of the features avoiding to extract them following manually designed templates. The parser was developed using the transition-based approach and it learns a classifier based on a neural network that chooses the correct transition using the arc-standard system and a greedy deterministic parsing algorithm. Each word w_i , PoS tag t_j and dependency type l_k is represented by its respective embedding vector e_i^w, e_j^t, e_k^l . The neural network chosen is a Multi-Layer Perceptron (MLP) with only one hidden layer. In the three layers data are represented by the features vector $[x^w, x^t, x^l]$. The x^w vector is composed of 18 embedding vectors for words in a given stack or buffer position of a configuration, so $x^w = [e_1^w; \dots; e_{18}^w]$. The same is true for the vector $x^t = [e_1^t; \dots; e_{18}^t]$, which includes 18 PoS embedding, and the $x^l = [e_1^l; \dots; e_{12}^l]$, consisting of 12 dependency type embedding vectors. The hidden layer h computes a linear combination of the input layer, a cubic activation function is applied ($g(x) = x^3$) and the result is merged into a softmax layer that predicts the transition with the highest probability:

$$\begin{aligned} h &= (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3 \\ p &= \text{softmax}(W_2 h) \end{aligned} \quad (1)$$

The cubic activation function, in place of classical activations such as hyperbolic tangent (tanh) and sigmoid function, is suitable for capturing the interactions between the three elements considered in model learning: words, PoS tags and dependency types. The training set $\mathcal{D}^{(\text{train})} = \{(c_i, t_i)\}$ consists of the configuration-transition pairs and does not use templates to transform the configuration. The parameters of the network are learned by minimizing the cross-entropy loss, using the l_2 regularization and the AdaGrad optimization algorithm with mini-batches.

In (Dyer et al. 2015) a transition-based parser based on the arc-standard system is proposed. This model try to learn the representation of the entire state of the parser, which is obtained from the representation of the buffer, the stack and the history of the actions taken by the parser. To represent the entire state it uses a technique that the authors called *LSTM stack*, based on Recurrent Neural Networks (RNN), more specifically on Long Short-Term Memories (LSTM) (Hochreiter and Schmidhuber 1997), and supports push and pop actions just like the elements of the parser configuration (stack and buffer). The idea is to represent the $S = \text{stack}(w_1, \dots, w_n)$ through the final state of the RNN applied to the word sequence w_1, \dots, w_n contained in the stack. The model uses three *LSTM stack* structures for each element of the configuration: one for the stack S , one for the buffer B and the last for the history A . At each step t the parser uses the representations s_t, b_t, a_t of the elements S, B and A to determine the transition to apply. The representation of the parser state at step t , called p_t , is defined as

$$p_t = \max\{0, W[s_t; b_t; a_t] + d\} \quad (2)$$

Each token x relative to any element of a configuration, given as input to the RNN, is represented by concatenating the embedding vector w of the word and the embedding vector t relative to its

PoS tag:

$$\mathbf{x} = \max\{0, \mathbf{V}[\mathbf{w}; \mathbf{t}] + \mathbf{b}\} \quad (3)$$

To represent the edges of the partial dependency tree in A it uses a vector \mathbf{c} obtained from the composition of the embedding vectors $\mathbf{h}, \mathbf{d}, \mathbf{r}$, which indicate the head, the dependent and the type of dependency, respectively:

$$\mathbf{c} = \tanh(\mathbf{U}[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{e}) \quad (4)$$

Given the representation of the correct sequence of transitions \mathbf{z} and the input phrase \mathbf{s} , the parser tries to minimize the negative conditional log-likelihood:

$$p(\mathbf{z}|\mathbf{s}) = - \sum_{t=1}^{|\mathbf{z}|} \log p(z_t|\mathbf{p}_t) \quad (5)$$

where $p(z_t|\mathbf{p}_t)$ is the probability of performing a specific transition z_t given the representation of the status \mathbf{p}_t . Network parameters are learned via the Stochastic Gradient Descent (SGD) optimization algorithm without mini-batches and applying a l_2 regularization factor. The adopted parsing algorithm can use a beam search.

In (Ballesteros, Dyer, and Smith 2015) a change to the representation of the equation (3) is proposed: the embedding vector \mathbf{w} of the word is replaced by a representation based on the single characters from which the word is composed, through the application of a bidirectional LSTM. Given a word, we call $\vec{\mathbf{w}}$ the vector of the final state of the RNN that reads the characters from left to right and $\overleftarrow{\mathbf{w}}$ the final state of the RNN that reads the characters in the opposite direction. The representation of a token \mathbf{x} in (3) is redefined as:

$$\mathbf{x} = \max\{0, \mathbf{V}[\vec{\mathbf{w}}; \overleftarrow{\mathbf{w}}; \mathbf{t}] + \mathbf{b}\} \quad (6)$$

This modification also introduces an additional transition to the arc-standard system that allows the production of dependency tree that are also non-projective.

(Kiperwasser and Goldberg 2016) follows the previous idea of representing the entire state of the parser using a two-way deep LSTM with k levels instead of the LSTM stack. The parser is developed in two versions: the first embodies a transition-based approach with an arc-hybrid and a dynamic oracle, while the second relies on a graph-based approach with an arc-factored model.

Given a sentence of n words w_1, \dots, w_n with the relative PoS tags t_1, \dots, t_n , each word w_i and PoS tag t_i are associated with the embedding vectors \mathbf{e}_i^w and \mathbf{e}_i^t , used to represent the input sequence $\mathbf{x}_1, \dots, \mathbf{x}_n$ and calculate the context \mathbf{v}_i as:

$$\begin{aligned} \mathbf{x}_i &= [\mathbf{e}_i^w; \mathbf{e}_i^t] \\ \mathbf{v}_i &= \text{BiLSTM}(\mathbf{x}_i) \end{aligned} \quad (7)$$

The \mathbf{v}_i features are used to obtain a representation ϕ that will be used as input for an MLP classifier, with only one hidden layer, which will calculate the score:

$$\text{MLP}(\phi(x)) = \mathbf{W}_2[\tanh(\mathbf{W}_2\phi(x) + \mathbf{b}_1)] + \mathbf{b}_2 \quad (8)$$

In the case of the transition-based model the MLP network learns the score of a transition given the $\phi(c)$ representation of a configuration c , obtained by combining the embedding vectors v_i associated with the first three words at the top of the stack and the first on the buffer:

$$\phi(c) = [v_{\sigma_3}; v_{\sigma_2}; v_{\sigma_1}; v_{\beta_1}] \quad (9)$$

In the case of the graph-based model the MLP classifier learns the score of each single arc (h, d) combining the embedding vector v_i of the head h and the dependent d :

$$\phi(h, d) = [v_h; v_d] \quad (10)$$

In the transition-based case they used a deterministic greedy parsing algorithm, while in the graph-based case the Eisner algorithm (described in (McDonald, Crammer, and Pereira 2006)) is used. Parameters are learned by minimizing the hinge loss using the Adam optimization algorithm.

In (Andor et al. 2016) a transition-based parser is proposed with an arc-standard system based on a network with a feed-forward architecture. Transition systems are characterized by a sequence of configurations c_1, \dots, c_j with their transitions t_1, \dots, t_j . The idea behind the model is to assume that there is a unique relationship between the sequence of transitions t_1, \dots, t_{j-1} and the state c_j . In other words, it is assumed that a state encodes the entire history of transitions. The goal is to learn, through the feed-forward network, the function $s(t_{1:j-1}, t_j)$ which calculates the score of the next valid transition t_j for c , given the sequence of previous transitions $t_{1:j-1}$ (which is assumed to encode the configuration c). Let \mathcal{T}_n be the set of valid transitions of length n , the model tries to find the solution to the optimization problem

$$\arg \max_{t_{1:n} \in \mathcal{T}_n} p_G(t_{1:n}) = \arg \max_{t_{1:n} \in \mathcal{T}_n} \sum_{j=1}^n s(t_{1:j-1}, t_j) \quad (11)$$

where p_G defines a Conditional Random Field (CRF) probability distribution for the transition sequence $t_{1:n}$:

$$p_G(t_{1:n}) = \frac{\exp \sum_{j=1}^n s(t_{1:j-1}, t_j)}{\sum_{t'_{1:n} \in \mathcal{T}_n} \exp \sum_{j=1}^n s(t'_{1:j-1}, t'_j)} \quad (12)$$

To approximate the $\arg \max$ function a beam search is used to make learning easier. The parameters are learned by minimizing the CRF loss and using the SGD optimization algorithm with momentum.

(Cheng et al. 2016) proposes an arc-factored graph-based parser that makes use of a bidirectional RNN with attention mechanism to analyze the sentence. The representation of each word w_i is obtained starting from the combination of the one-hot vectors e_i of the word attributes to which an LReLU activation function is applied:

$$x_i = \text{LReLU}[P(E^{\text{pos}} e_i^{\text{pos}} + E^{\text{form}} e_i^{\text{form}} + E^{\text{lemma}} e_i^{\text{lemma}} + \dots)] \quad (13)$$

$$\text{LReLU}(x) = \begin{cases} 0.1x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

The kind of RNN cell used for the components is the Gated Recurrent Unit (GRU) (Cho et al. 2014) with the LReLU activation function instead of the tanh. The state of the RNN is given by $\mathbf{h}_j = \text{GRU}(\mathbf{h}_{j-1}, \mathbf{x}_j)$, applied in both directions of the sentence so as to obtain \mathbf{h}_j^l and \mathbf{h}_j^r . In this model one-hot encoding is preferred rather than dense encoding because, in this way, it is possible to avoid establishing the dimension of the embeddings.

The parser consists of three components: memory, right-left queries, and left-right queries. Given a sentence $S = w_0 w_1 \dots w_n$ the parser constructs the $\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_n$ elements of the memory component by combining the states obtained from the application of the RNN on the sentence in both directions, so $\mathbf{m}_j = [\mathbf{h}_j^l, \mathbf{h}_j^r]$.

Each \mathbf{q}_t element of the query components is used to query the \mathbf{m}_j elements of the memory component and return a score $a_{t,j}$. The score indicates the weight of the attention relative to the arc composed of the word in position t (dependent) and that in position j (head) for $t = 1, \dots, n$ and $j = 0, \dots, n$, and it is calculated as:

$$a_{t,j} = \text{softmax}(\mathbf{V} \tanh(\mathbf{C}\mathbf{m}_j + \mathbf{D}\mathbf{q}_t)) \quad (14)$$

The authors defined the *soft* embedding of the elements of the memory component, similar to the attention mechanism, as $\tilde{\mathbf{m}}_t = \sum_{j=1}^n a_{t,j} \mathbf{m}_j$ and calculate the elements of the query components as $\mathbf{q}_t = \text{GRU}(\mathbf{q}_{t-1}, [\tilde{\mathbf{m}}_t \mathbf{x}_t])$. The representations of the two query components, in both directions, are passed to a MLP network with a single hidden layer that returns the probability of the head-dependent relationships as:

$$\mathbf{y}_t = \text{softmax}(\mathbf{U}[\tilde{\mathbf{m}}_t^l; \tilde{\mathbf{m}}_t^r] + \mathbf{W}[\mathbf{q}_t^l; \mathbf{q}_t^r]) \quad (15)$$

where $y_{t,1}, \dots, y_{t,m}$ are the probabilities of all m possible relationships. Analyzing all the head-dependent combinations one can implicitly capture graph-based information of a higher order than the first, even using an arc-factored model. The parsing algorithm used is the one proposed by (Chu and Liu 1965)/(Edmonds 1967) and the parameters of the network are learned by minimizing the cross-entropy loss using the Adam optimization algorithm.

(Dozat and Manning 2017; Dozat, Qi, and Manning 2017) proposed an arc-factored graph-based parser based on the one of (Kiperwasser and Goldberg 2016). The difference is that instead of using a similar MLP network they used *biaffine* layers. In the original approach the $\mathbf{v}_i \in \mathbb{R}^d$ state of the Equation (7) is used to calculate the s_i score of an arc by a linear transformation $s_i = \mathbf{W}\mathbf{v}_i + \mathbf{b}$ where $\mathbf{W} \in \mathbb{R}^{n \times d}$ and $\mathbf{b} \in \mathbb{R}^n$. In the biaffine architecture the transformation is obtained by introducing the product of two matrices $\mathbf{HW} \in \mathbb{R}^{d \times d}$ instead of the single \mathbf{W} matrix and the product $\mathbf{Hb} \in \mathbb{R}^d$ instead of bias \mathbf{b} . Before using biaffine transformation, the \mathbf{v}_i state is given as input to different MLPs with three levels in order to eliminate irrelevant information. The biaffine transformation is applied to the state generated by the MLP application

in order to predict the $\hat{y}_i^{(\text{arc})}$ head of the word i as:

$$\begin{aligned}
 \mathbf{h}_i^{(\text{arc-dep})} &= \text{MLP}^{(\text{arc-dep})}(\mathbf{v}_i) \\
 \mathbf{h}_i^{(\text{arc-head})} &= \text{MLP}^{(\text{arc-head})}(\mathbf{v}_i) \\
 \mathbf{s}_i^{(\text{arc})} &= \mathbf{H}^{(\text{arc-head})} \mathbf{W}^{(\text{arc})} \mathbf{h}_i^{(\text{arc-dep})} + \mathbf{H}^{(\text{arc-head})} \mathbf{b}^{(\text{arc})} \\
 \hat{y}_i^{(\text{arc})} &= \arg \max_j s_{ij}^{(\text{arc})}
 \end{aligned} \tag{16}$$

After predicting the head, this method selects the type of dependency $\hat{y}_i^{(\text{rel})}$ as follows:

$$\begin{aligned}
 \mathbf{h}_i^{(\text{rel-dep})} &= \text{MLP}^{(\text{rel-dep})}(\mathbf{v}_i) \\
 \mathbf{h}_i^{(\text{rel-head})} &= \text{MLP}^{(\text{rel-head})}(\mathbf{v}_i) \\
 \mathbf{s}_i^{(\text{rel})} &= \mathbf{h}_{\hat{y}_i^{(\text{arc})}}^{\top (\text{rel-head})} \mathbf{U}^{(\text{rel})} \mathbf{h}_i^{(\text{rel-dep})} + \mathbf{W}^{(\text{rel})} (\mathbf{h}_i^{(\text{rel-dep})} \oplus \mathbf{h}_{\hat{y}_i^{(\text{arc})}}^{(\text{rel-head})}) + \mathbf{b}^{(\text{rel})} \\
 \hat{y}_i^{(\text{rel})} &= \arg \max_j s_{ij}^{(\text{rel})}
 \end{aligned} \tag{17}$$

The chosen MST algorithm for parsing is Chu-Liu/Edmonds. The parameters are learned by training jointly the two biaffine classifiers and minimizing the sum of the cross-entropy loss.

In (Shi, Huang, and Lee 2017; Shi et al. 2017) the authors proposed a parser that combines a compact representation of the features of three different parsing paradigms: the two arc-hybrid and arc-eager systems of the transition-based approach and the arc-factored model for the graph-based approach. For each sentence the embedding vectors of each word are derived, through the use of a bidirectional LSTM network, starting from the character-level representation described in (Ballesteros, Dyer, and Smith 2015). The graph-based model learns the score of the edges following the deep biaffine architecture from (Dozat and Manning 2017) previously discussed. The two transition-based models share the same configurations and the score is calculated using a deduction system that learns the joint model, with initial axiom $[0, 1]$ and final state $[0, n + 1]$. The sequence with the highest score of the deduction system leading to the configuration $[0, n + 1]$ constitutes the predicted sequence of transitions. The parameters are learned by minimizing the hinge loss through the Adam optimization algorithm.

In (Nguyen, Dras, and Johnson 2017), a neural network model is proposed that can jointly learn both the PoS tagging and the graph-based arc-factored dependency parsing. The basic idea is that the more the PoS tags are accurate the better the performance of the parsing improves and, vice versa, the structure of the dependency tree can solve some ambiguity of PoS tags.

Given an input sentence w_1, \dots, w_n consisting of n words, the embedding of each word w_i is built by concatenating the word embedding vector e_{w_i} to the vector $e_{w_i}^c$ obtained by embedding its representation in characters as in (Ballesteros, Dyer, and Smith 2015):

$$\mathbf{e}_i = [\mathbf{e}_{w_i}, \mathbf{e}_{w_i}^c]. \tag{18}$$

The i -th word w_i is represented by the vector \mathbf{v}_i obtained as $\mathbf{v}_i = \text{BiLSTM}(\mathbf{e}_i)$. The score of the edge with head w_i and dependent w_j is calculated as:

$$\text{score}(w_i, w_j) = \text{MLP}([\mathbf{v}_{w_i}; \mathbf{v}_{w_j}]) \tag{19}$$

The loss function L_{arc} used to learn the parameters of the parsing task is the hinge loss. For the PoS tagging the tag sequence is represented by the state obtained by applying a bidirectional LSTM on the tag sequence and the loss function $L_{\text{pos}}(\hat{t}, t)$ to be minimized is the cross-entropy, where \hat{t} is the sequence of the predicted PoS tags and t the real one. The parser uses the Eisner MST parsing algorithm and the model parameters are learned by minimizing the sum of the two loss functions L_{pos} and L_{arc} through the Adam optimization algorithm.

It is worth mentioning the Italian dependency parser available in the package *spaCy*² based on DNN models. Unfortunately we could not include this parser into our evaluation because the available models have been trained on different corpora and in different experimental conditions and retraining new models would have required the production of a new parser using the *spaCy* API.

In Table 1 we summarised the fundamental characteristics for all the nine parsers considered in this study.

Table 1

All the neural parsers considered in this study with their fundamental features as well as their abbreviations used throughout the paper. In this table “Tb/Gb” means “Transition/Graph-based”, “arc-s/h/f” means “arc-standard/hybrid/factored” and “cle” indicates the Chu-Liu/Edmonds algorithm.

Parser Reference	Abbrev.	Method	Parsing
(Chen and Manning 2014)	CM14	Tb: arc-s	greedy
(Ballesteros, Dyer, and Smith 2015)	BA15	Tb: arc-s	beam search
(Kiperwasser and Goldberg 2016)	KG16:T	Tb: arc-h	greedy
(Kiperwasser and Goldberg 2016)	KG16:G	Gb: arc-f	eisner
(Andor et al. 2016)	AN16	Tb: arc-s	beam search
(Cheng et al. 2016)	CH16	Gb: arc-f	cle
(Dozat and Manning 2017)	DM17	Gb: arc-f	cle
(Shi, Huang, and Lee 2017; Shi et al. 2017)	SH17	Tb: arc-h	greedy-eager
(Nguyen, Dras, and Johnson 2017)	NG17	Gb: arc-f	eisner

3. Datasets

We set-up each parser using the data from the Italian Universal Dependencies (UD) treebanks, UD Italian 2.1 (general texts) and UD Italian PoSTWITA 2.2 (tweets). These treebanks are annotated following the Universal Dependencies v2 format (Nivre et al. 2016). Before proceeding with the experiments, all the treebanks were converted from the CoNLL-U format to the CoNLL-X format (Buchholz and Marsi 2006), using the conversion script made available in the official UD repository. This conversion was necessary because some parsers were developed previously at the definition of the CoNLL-U format and accept only the CoNLL-X format.

3.1 UD Italian 2.1

This corpus contains generic domain texts³. The UD Italian treebank was obtained by converting the corpus ISDT (Italian Stanford Dependency Treebank) from the Stanford Dependencies annotation scheme to the Universal Dependencies scheme, as described in (Attardi, Saletti,

² <https://spacy.io/>

³ <https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2515>

and Simi 2015). The ISDT was released for the first time during the dependency parsing task at EVALITA 2014 and was derived from the conversion of the MIDT corpus (Merged Italian Dependency Treebank) (Bosco, Montemagni, and Simi 2013). MIDT is the result of the merger and conversion of two previously existing dependency treebanks for the Italian language:

- The Turin University Treebank (TUT) (Bosco et al. 2000);
- The ISST-TANL treebank, initially released as ISST-CoNLL for the CoNLL 2007 shared task and developed from the Italian Syntactic-Semantic Treebank (ISST) (Montemagni et al. 2003).

The whole corpus is composed of 13,884 unique sentences, those already contained in the ISDT treebank plus other new sentences added after the conversion in the UD format. The subdivision of the entire corpus in Training, Development and Test sets is shown in Table 2.

Table 2
UD Italian 2.1 corpus splitting.

	Sentences	Words
Training set	12,838	270,703
Development set	564	11,908
Test set	482	10,417

3.2 UD Italian PoSTWITA 2.2

The second corpus we used contains social media texts⁴. The corpus, described in (Sanguinetti et al. 2018), consists of texts taken from the Twitter Italian platform. UD Italian PoSTWITA was created from a dataset used for the part-of-speech social media tagging in EVALITA 2016. The subdivision of the corpus is shown in Table 3.

Table 3
UD Italian PoSTWITA 2.2 corpus splitting.

	Sentences	Words
Training set	5,368	99,441
Development set	671	12,335
Test set	674	12,668

4. Neural Parsers Evaluation

For all parsers, we used the default settings for training, following the recommendation of the developers. Table 4 summarise the set up for each parser listing the values of each relevant hyperparameter.

⁴ https://github.com/UniversalDependencies/UD_Italian-PoSTWITA

Table 4

Values of the parsers hyperparameters: η is the learning rate; γ is the momentum factor; $\epsilon, \beta_1, \beta_2$ are the parameters for the Adam optimiser; b is the size of the mini-batch; p is the probability of dropout; h is the width of the MLP hidden levels; k is the number of RNN levels; l the size of the RNN; λ is the weight of the regularization term.

Parser	Hyperparameters
CM14	$\eta = 0.01$, $\epsilon = 10^{-6}$, $b = 10.000$, $p = 0.5$, $h = 200$, iter = 20.000, $\lambda = 10^{-8}$
BA15	$\eta = 0.1$, $k = 2$, $l = 100$, iter = 5.500, $\lambda = 10^{-6}$
KG16	$\eta = 0.1$, $\epsilon = 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $k = 2$, $l = 125$, $h = 100$, epochs = 30, $p = 0.5$
AN16	$\eta = 0.02$, $\gamma = 0.9$, $h_1 = h_2 = 512$, $b = 8$, beam = 16, epochs = 10
CH16	$\eta = 0.0004$, $h = 368$, $b = 1$
DM17	$\eta = 0.002$, $\beta_1 = \beta_2 = 0.9$, $\epsilon = 10^{-12}$, $k = 3$, $l = 300$, $h = 100$, $b = 5.000$, iter = 25.000, $p = 0.33$
SH17	$\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.9999$, $\epsilon = 10^{-8}$, $k = 2$, $l = 256$, $h = 128$, $b = 50$, epochs = 20
NG17	$\eta = 0.1$, $\epsilon = 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $k = 2$, $l = 128$, $h = 100$, epochs = 30

The experiments were organized in three different setups:

- **setup0**: we trained and tested the parsing models on the generic Italian treebank (UD Italian 2.1 dataset). This setup will be useful to compare the performance of the parsers trained on generic texts with those obtained starting from texts coming from social media;
- **setup1**: we trained and tested the parsing models on a treebank in Italian language consisting solely of social media domain texts (UD Italian PoSTWITA 2.2);
- **setup2**: we trained the parsing models by joining the train and validation subsets of the the previous setups together and keeping the test set for setup1.

Evaluation results were obtained using the *Evaluation tool* software DEPENDABLE⁵, described in (Choi, Tetreault, and Stent 2015), based on the standard evaluation script `eval.pl` of the CoNLL-X Shared Task.

After the influential paper from (Reimers and Gurevych 2017) it is clear to the community that reporting a single score for each DNN training session could be heavily affected by the system initialisation point and we should instead report the mean and standard deviation of various runs with the same setting in order to get a more accurate picture of the real systems performance and make more reliable comparisons between them.

For each parser we evaluated five different instances; counts include punctuation and report the mean and standard deviation of the values obtained in the different runs for each setup. On all models, the statistical significance level was calculated using DEPENDABLE and applying the McNemar statistical test.

⁵ Available online: <https://github.com/emorynlp/dependable>

Table 5 shows the parsers' performance on the test set for the three setups described above executing the training/validation/test cycle for 5 times.

EvalITA 2014 results reported the best score that establishes the state of the art for the dependency parsing in Italian: UAS 93.55% and LAS 88.76%⁶. The corpus used for setting up the systems participating to the task was the ISDT (Italian Stanford Dependency Treebank), the same corpus from which UD Italian was created. The UAS value for the best parser of experiments on UD Italian 2.1 (setup0) slightly exceeds the state of the art established in EvalITA 2014, while the LAS is 91.84%, 3.08% higher than the best parser in EvalITA 2014. However, it should be noted that despite being built on the same corpus, the UD Italian does not coincide exactly with the ISDT, so the comparison is not completely fair since the two treebanks have differences in their construction as an annotation scheme and sentences added and corrected in the UD Italian that do not appear in ISDT. In fact, in (Attardi, Saletti, and Simi 2015) it is shown how the change from ISDT to UD Italian 1.2 generates better performance on experiments conducted with statistical parsers. The results in the Attardi's paper are the only evaluations published in the literature on the UD Italian treebank, in version 1.2. Among the experiments conducted with statistical parsers the best result on UD Italian 1.2 is the Mate parser (Bohnet 2010; Bohnet and Kuhn 2012) with UAS evaluation 92.47% and LAS 90.22%. Even in this case we cannot make a complete comparison with the results obtained using UD Italian 2.1⁷. Notwithstanding that UD Italian 1.2 and UD Italian 2.1 are similar, they do not completely match; however, the evaluations obtained from the parser from (Dozat and Manning 2017) trained on UD Italian 2.1 are superior to those of Mate on UD Italian 1.2 in both UAS and LAS.

The results on setup1 are much lower when compared with those of setup0, but this is reasonable considering that the UD PoSTWITA treebank 2.2 is much smaller than the UD Italian 2.1 and that the syntactic analysis of tweets is linguistically more difficult than standard Italian. On the other hand, if we consider the models learned from the union of the two treebanks, we can see that there is still room for improvement with an increase of $\sim 1.7\%$ in UAS and $\sim 2\%$ in LAS, compared to the models learned using only the UD Italian PoSTWITA 2.2. From this we can deduce that to have good performance within a domain, specifically for social media texts, it is essential to use as much data as possible, including domain data, to train the parsing models. To further support this hypothesis, it can be noted that joining the two treebanks, UD Italian 2.1 and UD Italian PoSTWITA 2.2, we were able to get a nice performance improvement.

In any setup the DM17 parser exhibits the best performance, notably very high for general Italian. As we can expect, the performance on setup1 were much lower than that for setup0 due to the intrinsic difficulties of parsing tweets and to the scarcity of annotated tweets for training. Joining the two datasets in the setup2 allowed to get a relevant gain in parsing tweets even if we added out-of-domain data. For these reasons, for all the following experiments, we abandoned the setup1 because it seemed more relevant to use the joined data (setup2) and compare them to setup0.

5. An Ensemble of Neural Parsers

The DEPENDABLE tool in (Choi, Tetreault, and Stent 2015) is able to compute ensemble upper bound performance assuming that, given the parsers outputs, the best tree can be identified by an oracle "MACRO" (*MA*), or that the best arc can be identified by another oracle "MICRO" (*mi*). Table 6 shows that, by applying these oracles, we have plenty of space to improve the

⁶ http://www.evalita.it/2014/tasks/dep_par4IE

⁷ For changes between versions you can see the Changelog of the UD Italian corpus:

https://github.com/UniversalDependencies/UD_Italian-ISDT

Table 5

Mean/standard deviation of UAS/LAS for each parser and for the three different setups by repeating the experiments 5 times. All the results are statistically significant ($p < 0.05$) and the best values are showed in boldface.

setup0				
	Valid. Ita		Test Ita	
	UAS	LAS	UAS	LAS
CM14	88.20/0.18	85.46/0.14	89.33/0.17	86.85/0.22
BA15	91.15/0.11	88.55/0.23	91.57/0.38	89.15/0.33
KG16:T	91.17/0.29	88.42/0.24	91.21/0.33	88.72/0.24
KG16:G	91.85/0.27	89.23/0.31	92.04/0.18	89.65/0.10
AN16	85.52/0.34	77.67/0.30	87.70/0.31	79.48/0.24
CH16	92.42/0.00	89.60/0.00	92.82/0.00	90.26/0.00
DM17	93.37/0.27	91.37/0.24	93.72/0.14	91.84/0.18
SH17	89.67/0.24	85.05/0.24	89.89/0.29	84.55/0.30
NG17	90.37/0.12	87.19/0.21	90.67/0.15	87.58/0.11
setup1				
	Valid. PoSTW		Test PoSTW	
	UAS	LAS	UAS	LAS
CM14	81.03/0.17	75.24/0.30	81.50/0.28	76.07/0.17
BA15	83.44/0.20	77.70/0.25	84.06/0.38	78.64/0.44
KG16:T	77.38/0.14	68.81/0.25	77.41/0.43	69.13/0.43
KG16:G	78.81/0.23	70.14/0.33	78.78/0.44	70.52/0.51
AN16	77.74/0.25	66.63/0.16	77.78/0.33	67.21/0.30
CH16	84.78/0.00	78.51/0.00	86.12/0.00	79.89/0.00
DM17	85.01/0.16	78.80/0.09	86.26/0.16	80.40/0.19
SH17	80.52/0.18	73.71/0.14	81.11/0.29	74.53/0.26
NG17	82.02/0.11	75.20/0.24	82.74/0.39	76.22/0.41
setup2				
	Valid. Ita+PoSTW		Test PoSTW	
	UAS	LAS	UAS	LAS
CM14	85.52/0.13	81.51/0.05	82.62/0.24	77.45/0.23
BA15	87.85/0.13	83.80/0.12	85.15/0.29	80.12/0.27
KG16:T	83.89/0.23	77.77/0.26	80.47/0.36	72.92/0.46
KG16:G	84.70/0.14	78.41/0.14	81.41/0.37	73.49/0.19
AN16	82.95/0.33	73.46/0.37	79.81/0.27	69.19/0.19
CH16	89.16/0.00	84.56/0.00	86.85/0.00	80.93/0.00
DM17	89.72/0.10	85.85/0.13	87.22/0.24	81.65/0.21
SH17	85.85/0.36	80.00/0.39	83.12/0.50	76.38/0.38
NG17	86.81/0.04	82.13/0.09	84.09/0.07	78.02/0.11

performance by building some kind of ensemble system able to cleverly choose the correct information from the different parsers outputs and combine them improving the final solution. This observation motivates our proposal.

Table 6

Results obtained by building an ensemble system based on the oracles mi e MA computed by the DEPENDABLE tool considering all parsers outputs.

	Validation		Test	
	UAS	LAS	UAS	LAS
setup0				
mi	98.30%	97.82%	98.08%	97.72%
MA	96.62%	95.10%	96.31%	94.82%
setup2				
mi	97.08%	96.02%	96.32%	94.73%
MA	94.62%	91.29%	93.27%	88.50%

To combine the parser outputs we tested three ensemble schemas proposed in literature: voting, reparsing and distilling. The next three subsections discuss these techniques applied to our problem and present the obtained results.

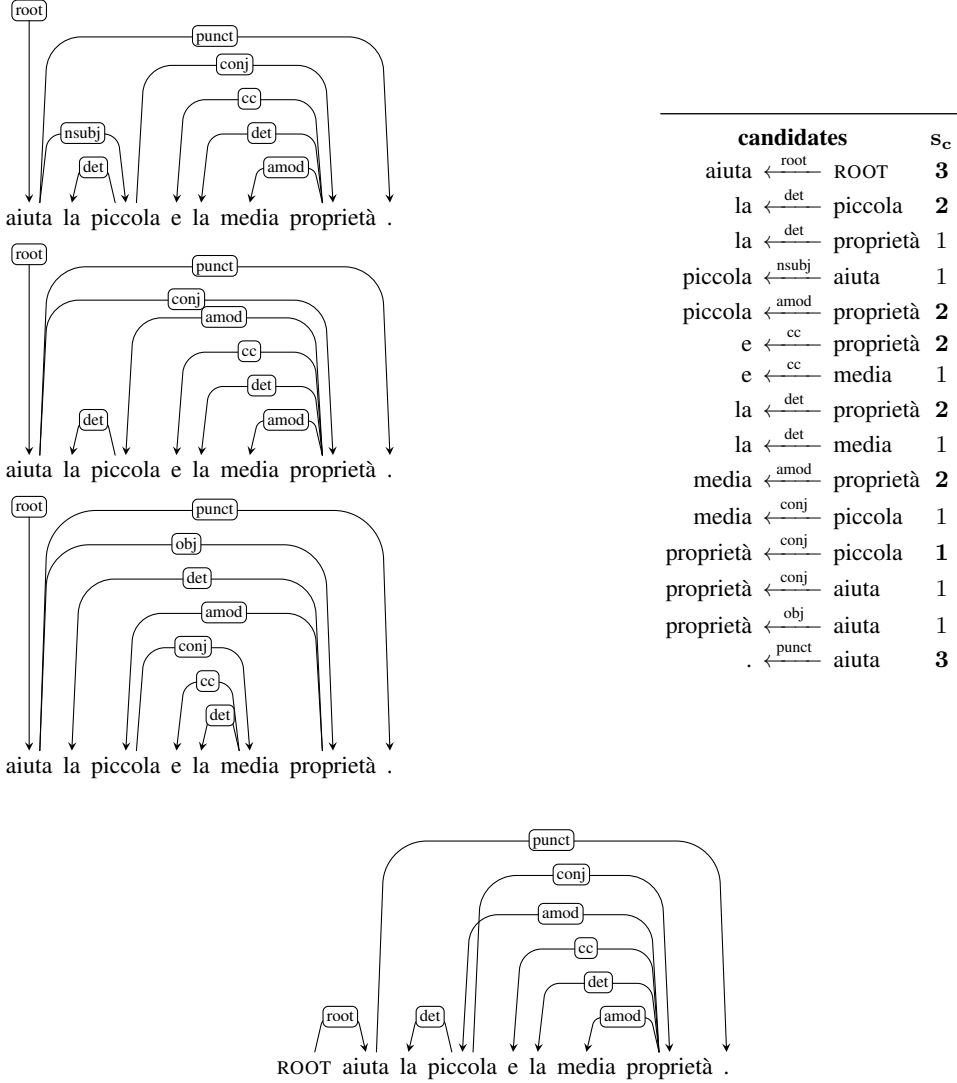
5.1 Voting

The voting technique was proposed for the first time in (Zeman and Žabokrtský 2005). Given the sentence $S = w_1 w_2 \dots w_n$, each of m basic parsers contributes to the defined ensemble by assigning one score to each candidate dependency relation (w_i, r, w_j) , with $0 \leq i, j \leq n$ and $i \neq j$, where w_i is the dependent, w_j is the head and r is the type of the dependency relation. Dependency relations are contained among those available in the m trees predicted by parsers. In the example of Figure 1 we consider three dependency trees (shown in the upper left) produced by three different parsers. Starting from these, each individual distinct arc becomes an arc classified in the list of candidate arcs. All the candidate arcs, therefore, appear in at least one of the starting dependency trees. The s_c score is the number of votes of each candidate arc, calculated as the number of times that it is part of an individual dependency tree. After passing the list of candidates with their scores, for each word contained in the sentence the arc that has the maximum score is accepted and, in the event of a tie, the arc from the first parser is chosen. In the example of Figure 1 there is a tie for the word *proprietà* whose three candidate arcs all have $s_c = 1$. This voting strategy is known as *majority*.

Since there is no restriction on the structure of the tree, the tree generated by the majority strategy has no guarantee to be well formed. As you can see in Figure 1, starting from three well-formed dependency trees a new dependency tree has been composed that is not well formed. To avoid this problem it is possible to use the strategy of *switching* which consists in checking if the final tree obtained through the majority strategy is well formed and, if not, replace it entirely with the dependency tree produced by the first parser. In order to build a feasible ensemble applying these techniques, we must consider at least the outputs of three different parsers.

As a preliminary analysis we try to capture and measure the diversity, or equivalently the similarity, of the available parsers. The basic idea is to understand how many times the parsers agree in predicting a certain relationship, this measure their agreement.

Consider two dependency trees $T = (V, A)$ and $\tilde{T} = (V, \tilde{A})$ for the same sentence $S = w_0 w_1 w_2 \dots w_n$ where the set of nodes V (the words in the sentence) are common to both trees and the set of edges A and \tilde{A} can be different. It is possible to define the *agreement* between two

**Figure 1**

From the dependency trees obtained from three basic parsers (left) we establish the candidate arcs (right) from which we can generate the final dependency (below). The final tree is not well formed because it is not completely connected and presents the cycle *piccola* →_{conj} *proprietà* →_{amod} *piccola*.

dependency trees as:

$$\text{agreement}(T, \tilde{T}) = \frac{1}{|A|} \sum_{\substack{(w,r,h) \in A \\ (w,\tilde{r},\tilde{h}) \in \tilde{A}}} \begin{cases} 1 & \text{if } h = \tilde{h} \text{ and } r = \tilde{r} \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

where (w, r, h) and $(w, \tilde{r}, \tilde{h})$ are the dependency relations in the T and \tilde{T} tree for each word $w \in \{w_1, \dots, w_n\}$. From the definition it follows that the agreement is a symmetrical measure,

so we have that $\text{agreement}(T, \tilde{T}) = \text{agreement}(\tilde{T}, T)$. In other words the agreement calculates the percentage of the dependency relationships between two trees that have been labeled in the same way. By extending the definition of agreements on a set of trees computing the average on the whole treebank, we can calculate the agreement between the parsers using their predictions. In Table 7 the agreement for each pair of parsers using the models learned in setup0 is reported, while in Table 8 those in setup2.

Table 7

Agreement calculated on the development set starting from the predictions of the models learned in the setup0 (UD Italian 2.1).

	BA15	KG16:T	KG16:G	AN16	CH16	DM17	SH17	NG17
CM14	87.59%	87.88%	87.81%	81.19%	88.11%	88.30%	80.67%	85.74%
BA15		88.97%	90.26%	82.11%	89.85%	90.59%	82.85%	87.64%
KG16:T			91.25%	82.27%	90.25%	91.02%	82.61%	88.01%
KG16:G				82.85%	90.89%	91.92%	83.30%	88.81%
AN16					82.48%	83.56%	78.83%	84.32%
CH16						92.38%	83.98%	88.66%
DM17							85.38%	90.27%
SH17								83.70%

Table 8

Agreement calculated on the development set starting from the predictions of the models learned in the setup2 (UD Italian 2.1+PoSTWITA 2.2).

	BA15	KG16:T	KG16:G	AN16	CH16	DM17	SH17	NG17
CM14	83.79%	77.13%	76.95%	77.74%	83.73%	83.60%	74.33%	80.56%
BA15		78.34%	78.10%	78.65%	84.93%	85.58%	76.18%	82.98%
KG16:T			80.15%	77.46%	79.02%	79.85%	72.95%	80.22%
KG16:G				77.54%	79.02%	80.68%	72.90%	80.79%
AN16					78.76%	79.50%	72.34%	80.16%
CH16						87.19%	76.70%	83.66%
DM17							77.45%	84.60%
SH17								76.31%

The agreement depends largely on the performance of a parser because it is related to the LAS metric, which can be defined as $\text{agreement}(T, G)$, where G is the gold standard dependency tree.

Even if the majority strategy could generate ill-formed parses, it will be considered to provide a comparison with the switching strategy. In some cases even having a ill-formed tree, but with a good accuracy, could be useful in some processes where a manual correction of the dependency tree is provided, as happens, for example, in the semi-automatic annotation of a treebank. For this reason we developed some experiments considering both the majority and the switching strategy.

For the experiments the following voter configurations were analyzed:

- (DM17 + CH16 + BA15) considers the three best parsers, which are (Dozat and Manning 2017), (Cheng et al. 2016) and (Ballesteros, Dyer, and Smith 2015);

- (AN16 + CM14 + SH17) considers the worst three parsers, which are (Andor et al. 2016), (Chen and Manning 2014) and (Shi, Huang, and Lee 2017);
- (DM17 + CM14 + SH17) considers the best parser, (Dozat and Manning 2017), combining it with those that have minor agreements, (Chen and Manning 2014) and (Shi, Huang, and Lee 2017);
- (AN17 + ALL) considers all parsers with (Andor et al. 2016) as the first parser;
- (DM17 + ALL) considers all parsers with (Dozat and Manning 2017) as the first parser.

Table 9 shows the performance of the ensembles built on the best results on validation set obtained in the 5 training/test cycles considering both setup0 and setup2. Table 11 reports the number of ill-formed trees for the majority strategy, while Table 10 reports the number of cases when the ensemble combination output differs from the baseline, including both labeled (L) and unlabeled (U) outputs. For the best results (DM17+ALL) the difference on setup0 and setup2 is about 4%.

The results of the voting approach reported in Table 9 shows that the majority strategy is slightly better than the switching strategy, although it must be taken into account that there might be ill-formed dependency trees for the former. The percentage of ill-formed trees on validation/test set vary from a minimum of 2% to a maximum of 8%. For this reasons the majority strategy should be used when it is followed by a manual correction phase. The switching strategy performs well if the first parser of voters is one of the best parsers, in fact the combinations AN16+ALL and AN16+CM14+SH17 have worst performance than the counterparts using the best parser (DM17) as the first voter. Overall, the highest performance is achieved using all parsers together with DM17 as the first voter. For setup0 the increases were +0.19% in UAS e +0.38% in LAS, while in setup2 are +0.92% in UAS e +2.47% in LAS with respect to the best single parser (again DM17).

5.2 Reparsing

A different approach for building ensemble systems, proposed in (Sagae and Lavie 2006), is *reparsing*. In this approach we try to overcome the limitation of the majority strategy, related to the possibility of generating trees that are not well formed, exploiting the same methodologies used in some parsing algorithms. This technique builds a directed graph with all the distinct candidate arcs provided by the single parsers, where the edge weights are obtained from a specific voting algorithm. Finally, classical parsing techniques are applied to the graph using an MST algorithm that generates a well-formed dependency tree.

We will consider two types of MST algorithms for the experiments: the first is the Chu-Liu/Edmonds algorithm, which searches the entire space of non-projective dependency trees, so the generated tree can also be non-projective. For this type of approach we can also use different techniques to assign the vote, which will influence the weight attributed to the arc in the graph. We will use the three different voting weighting methods proposed in (Hall et al. 2007):

- w_2 : equally weighted;
- w_3 : weighted according to the total labeled accuracy on the validation set;
- w_4 : weighted according to labeled accuracy per coarse grained PoS tag on the validation set.

Table 9

Results of ensembles using switching and majority approaches on the best models in setup0 and setup2. The baseline is defined by the best results of DM17.

setup0				
Voters/Strategy	Validation		Test	
	UAS	LAS	UAS	LAS
DM17+CH16+BA15/maj.	94.20%	92.27%	93.77%	92.13%
DM17+CH16+BA15/swi.	94.11%	92.16%	93.79%	92.14%
AN16+CM14+SH17/maj.	90.43%	87.96%	91.03%	88.47%
AN16+CM14+SH17/swi.	89.44%	86.77%	90.17%	87.43%
DM17+CM14+SH17/maj.	93.84%	92.03%	93.82%	92.27%
DM17+CM14+SH17/swi.	93.76%	91.94%	93.82%	92.25%
AN16+ALL/maj.	94.37%	92.65%	93.83%	92.27%
AN16+ALL/swi.	93.99%	92.15%	93.43%	91.73%
DM17+ALL/maj.	94.42%	92.67%	93.94%	92.41%
DM17+ALL/swi.	94.38%	92.60%	93.91%	92.37%
DM17 (baseline)	93.74%	91.66%	93.75%	92.03%

setup2				
Voters/Strategy	Validation		Test	
	UAS	LAS	UAS	LAS
DM17+CH16+BA15/maj.	90.57%	87.16%	88.21%	83.64%
DM17+CH16+BA15/swi.	90.51%	87.10%	88.13%	83.51%
AN16+CM14+SH17/maj.	86.90%	83.60%	84.09%	79.78%
AN16+CM14+SH17/swi.	86.01%	82.50%	82.58%	77.94%
DM17+CM14+SH17/maj.	90.35%	87.21%	88.07%	83.64%
DM17+CM14+SH17/swi.	90.27%	87.11%	87.99%	83.52%
AN16+ALL/maj.	90.30%	87.26%	88.36%	84.13%
AN16+ALL/swi.	89.70%	86.45%	87.46%	83.06%
DM17+ALL/maj.	90.64%	87.60%	88.51%	84.42%
DM17+ALL/swi.	90.65%	87.62%	88.50%	84.20%
DM17 (baseline)	89.82%	85.96%	87.59%	81.95%

As noted by (McDonald et al. 2005), searching the entire space of non-projective dependency trees can sometimes be counterproductive. Although some languages allow non-projective relationships, they are still mostly projective. So, looking through all the non-projective trees, we run the risk of finding dependency trees that are not desirable even if they are well formed. For example, the training set of the UD Italian corpus contains 564 non-projective dependency tree on the entire corpus of 12838 dependency tree ($\sim 4.4\%$). For this reason we consider a second MST algorithm that searches only the projective dependency tree, the Eisner algorithm.

For the reparsing experiments we will consider only the best three parsers (DM17 + CH16 + BA15) and ALL parsers (the order in this case is not important).

Table 12 shows the performance of the ensembles built on the best results on validation set obtained in the 5 training/test cycles considering both setup0 and setup2, while Table 13 reports the number of cases when the ensemble combination output differs from the baseline, including both labeled (L) and unlabeled (U) outputs.

Table 10

Numbers of cases when there is a different output between the ensemble systems, using switching and majority, and the baseline DM17.

setup0				
Voters/Strategy	Validation		Test	
	/11.908		/10.417	
	U	L	U	L
DM17+CH16+BA15/maj.	208	61	188	46
DM17+CH16+BA15/swi.	192	52	175	39
AN16+CM14+SH17/maj.	1.006	424	783	336
AN16+CM14+SH17/swi.	1.130	489	870	371
DM17+CM14+SH17/maj.	170	37	139	15
DM17+CM14+SH17/swi.	157	33	129	13
AN16+ALL/maj.	382	126	328	105
AN16+ALL/swi.	460	164	386	133
DM17+ALL/maj.	356	117	282	81
DM17+ALL/swi.	312	97	255	72

setup2				
Voters/Strategy	Validation		Test	
	/24.243		/12.668	
	U	L	U	L
DM17+CH16+BA15/maj.	597	219	470	213
DM17+CH16+BA15/swi.	521	185	394	172
AN16+CM14+SH17/maj.	2.757	1.329	1.805	941
AN16+CM14+SH17/swi.	2.976	1.429	1.986	1.033
DM17+CM14+SH17/maj.	490	140	337	93
DM17+CM14+SH17/swi.	453	121	300	73
AN16+ALL/maj.	1.377	624	897	440
AN16+ALL/swi.	1.610	741	1.063	534
DM17+ALL/maj.	1.156	502	784	378
DM17+ALL/swi.	920	374	614	280

Table 11

Number of ill-formed trees obtained by using the majority strategy for both setups.

Voters	setup0		setup2	
	Valid.	Test	Valid.	Test
	/564	/482	/1235	/674
DM17+CH16+BA15	9	7	31	31
AN16+CM14+SH17	45	25	88	77
DM17+CM14+SH17	6	6	19	23
AN16+ALL	18	17	73	63
DM17+ALL	17	11	75	57

The results of the reparsing approach reported in Table 12 shows that the Chu-Liu/Edmonds algorithm is slightly better than the Eisner algorithm. In this case, the choice of the strategy that will be used depends on our decision of requesting the presence or the absence of non-projectivity. The percentage of non-projective dependency trees on valid/test set for Chu-Liu/Edmonds vary from a minimum of 7% to a maximum of 12% compared with the average for the Italian corpora of 4%. Overall, the highest performance are achieved using Chu-Liu/Edmonds algorithm. For setup0 the increases are +0.25% in UAS and +0.45% in LAS, while in setup2 are +0.77% in UAS and +2.30% in LAS with respect to the best single parser (DM17).

Table 12

Results of the proposed ensembles built by using reparsing approaches on the best models in setup0 and setup2. The baseline is again defined by the best results of DM17.

setup0				
Voters/Strategy	Validation		Test	
	UAS	LAS	UAS	LAS
DM17+CH16+BA15/cle-w2	93.82%	91.85%	93.54%	91.83%
DM17+CH16+BA15/cle-w3	93.89%	91.82%	93.78%	92.06%
DM17+CH16+BA15/cle-w4	94.20%	92.28%	93.72%	92.04%
DM17+CH16+BA15/eisner	94.05%	92.05%	93.46%	91.78%
ALL/cle-w2	94.31%	92.53%	93.85%	92.23%
ALL/cle-w3	94.16%	92.41%	94.00%	92.48%
ALL/cle-w4	94.29%	92.58%	93.95%	92.38%
ALL/eisner	94.31%	92.53%	93.95%	92.35%
DM17 (baseline)	93.74%	91.66%	93.75%	92.03%
setup2				
Voters/Strategy	Validation		Test	
	UAS	LAS	UAS	LAS
DM17+CH16+BA15/cle-w2	90.33%	86.95%	87.69%	83.31%
DM17+CH16+BA15/cle-w3	89.82%	85.96%	87.59%	81.95%
DM17+CH16+BA15/cle-w4	90.41%	86.99%	87.94%	83.32%
DM17+CH16+BA15/eisner	90.50%	87.05%	88.04%	83.51%
ALL/cle-w2	90.52%	87.53%	88.36%	84.25%
ALL/cle-w3	89.90%	86.75%	87.79%	83.54%
ALL/cle-w4	90.42%	87.46%	88.19%	84.11%
ALL/eisner	90.45%	87.41%	88.31%	84.08%
DM17 (baseline)	89.82%	85.96%	87.59%	81.95%

5.3 Distilling

Recently, in (Kuncoro et al. 2016), an approach has been proposed for the construction of a voting-based ensemble, later defined as *distilling*, which allows the training of a single model starting from different parsers independently trained. The technique consists of learning a single parsing model from a cost matrix based on the votes of the m parser, using a particular cost function that considers the uncertainty of the prediction. The basic idea is that the disagreement among parsers can be a signal that the relation in question is ambiguous. Training a distilling model takes a long time, but it has the advantage of creating a final model that does not need to query the individual parsers from which it was built. In the ensembles considered so far the

Table 13

Numbers of cases when there is a different output between the ensemble systems, using reparsing approaches, and the baseline DM17.

setup0				
Voters/Strategy	Validation		Test	
	/11.908		/10.417	
	U	L	U	L
DM17+CH16+BA15/cle-w2	360	129	307	90
DM17+CH16+BA15/cle-w3	96	0	89	1
DM17+CH16+BA15/cle-w4	267	76	247	52
DM17+CH16+BA15/eisner	375	130	327	103
ALL/cle-w2	400	131	333	103
ALL/cle-w3	351	108	299	79
ALL/cle-w4	383	126	307	87
ALL/eisner	411	133	333	106

setup2				
Voters/Strategy	Validation		Test	
	/24.243		/12.668	
	U	L	U	L
DM17+CH16+BA15/cle-w2	1.056	496	800	424
DM17+CH16+BA15/cle-w3	0	0	0	0
DM17+CH16+BA15/cle-w4	603	264	491	236
DM17+CH16+BA15/eisner	1.047	443	789	376
ALL/cle-w2	1.347	599	882	417
ALL/cle-w3	1.261	537	804	363
ALL/cle-w4	1.274	576	822	389
ALL/eisner	1.367	607	916	436

composition time of the final dependency tree is a function of the sum of the parsing times of the m parser used to build the ensemble. In the case of distilling, the model is created only once starting from the m parser that will no longer be used.

The results of the distilling strategy using all the available parsers are reported in Table 14: unlike the previous ensemble proposals this technique exhibits worse outcomes which score below the baseline (DM17).

Table 14

Results of distilling approach on the best models in setup0 and setup2. In brackets are reported the differences between the distilled models, built by considering all parsers, and the best results of DM17, as baseline.

Setup	UAS	LAS
<i>setup0</i>	92.50% (−1.25%)	89.93% (−2.10%)
<i>setup2</i>	86.73% (−0.86%)	81.39% (−0.56%)

6. Discussion and Conclusions

For a clear comparison between the different ensemble techniques tested in our experiments, the best results have been summarised in Table 15.

Table 15

The table summarises the best results obtained for the different ensemble strategies on the test set both for setup0 and setup2. Improvements with respect to the baseline (DM17) are shown in brackets. The largest improvements are marked in bold for both setups.

setup0		
strategy	UAS	LAS
DM17+ALL/majority	93.94% (+0.19%)	92.41% (+0.38%)
DM17+ALL/switching	93.91% (+0.16%)	92.37% (+0.34%)
ALL/cle-w3	94.00% (+0.25%)	92.48% (+0.45%)
ALL/eisner	93.95% (+0.20%)	92.35% (+0.32%)
ALL/distilling	92.50% (−1.25%)	89.93% (−2.10%)
setup2		
strategy	UAS	LAS
DM17+ALL/majority	88.51% (+0.92%)	84.42% (+2.47%)
DM17+ALL/switching	88.50% (+0.91%)	84.20% (+2.25%)
ALL/cle-w2	88.36% (+0.77%)	84.25% (+2.30%)
ALL/eisner	88.31% (+0.72%)	84.08% (+2.13%)
ALL/distilling	86.73% (−0.86%)	81.39% (−0.56%)

The distilling approach shows lower results than the baseline and, in general, lower than the other methods. The best results are obtained through the majority approach which brings with it the problem of not ensuring that the tree generated is well formed. The difference between the two setups is that in setup0 the improvements do not exceed 0.5%, while in setup2 they obtained a 2.5% of gain in LAS. As for reparsing methods, both in setup0 and in setup2 the Chu-Liu/Edmonds algorithm turns out to be slightly better than Eisner’s procedure. All the techniques return more or less the same improvements, with oscillations of a few decimal points.

Therefore, the choice of the strategy is due, in part, to the properties we desire in the final tree. If we are not interested in the correctness of the tree structure, because it will be manually corrected, we can safely use the majority technique. If instead we want a correct tree and we have a parser exhibiting good performance, we could use the switching technique selecting it as the first parser. If the language can contain many non-projective structures, we might want to direct our choice on reparsing with the Chu-Liu/Edmonds algorithm or, if we prefer to get projective trees, on the Eisner algorithm. As far as distilling is concerned, it has not proved to be a good method to use for models obtained from different parsers but, as the author suggests, it could be useful when we consider several models obtained from different instances of the same parser.

We have studied the performance of some neural dependency parsers on generic and social media domain. Using the predictions of each single parser we combined the best outcomes to improve the performance in various ways. The ensemble models give an improvement of $\sim 1\%$ in UAS and $\sim 2.5\%$ in LAS in the mixed setup (2).

The improvement of LAS is, in most cases, at least twice the value of UAS. This could mean that ensemble models catch with better precision the type of dependency relations rather than head-dependent relations, since the candidates relations are taken from existing dependency

trees and not generated from scratch. This consideration needs, however, further studies in order to be verified.

All the proposed ensemble strategies, except for distilling, perform more or less in the same way, therefore the choice of the strategy to use is due, in part, to the properties that we want to obtain on the combined dependency tree.

Our proposal was inspired by the work of (Mazzei 2015). Unlike from his study, we use a larger set of state-of-the-art parsers, all based on neural networks, in order to gain more diversity among the models used in the ensembles; furthermore we have experimented the distilling strategy and the Eisner reparsing algorithm and we built ensembles on larger datasets using both generic and social media texts.

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

- Alicante, Anita, Cristina Bosco, Anna Corazza, and Alberto Lavelli. 2015. Evaluating Italian Parsing Across Syntactic Formalisms and Annotation Schemes. In Roberto Basili, Cristina Bosco, Rodolfo Delmonte, Alessandro Moschitti, and Maria Simi, editors, *Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project*. Springer International Publishing, Cham, pages 135–159.
- Andor, Daniel, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August. ACL.
- Attardi, Giuseppe, Simone Saletti, and Maria Simi. 2015. Evolution of Italian Treebank and Dependency Parsing towards Universal Dependencies. In *Proceedings of the Second Italian Conference on Computational Linguistics CLiC-it 2015: 3-4 December 2015, 2015*. Torino: Accademia University Press.
- Ballesteros, Miguel, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. ACL.
- Bohnet, Bernd. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 89–97, Beijing, China, August 23–27.
- Bohnet, Bernd and Jonas Kuhn. 2012. The Best of Both Worlds – A Graph-based Completion Model for Transition-based Parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 77–87, April 23–27, Avignon, France.
- Bosco, Cristina, Felice Dell’Orletta, Simonetta Montemagni, Manuela Sanguinetti, and Maria Simi. 2014. The EVALITA 2014 Dependency Parsing task. In *Proceedings of the Fourth International Workshop EVALITA 2014*, pages 1–8, Pisa, Italy, December.
- Bosco, Cristina, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 2000. Building a Treebank for Italian: a Data-driven Annotation Schema. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000*, Athens, Greece, 31 May - June 2.
- Bosco, Cristina and Alessandro Mazzei. 2011. The EVALITA 2011 Parsing Task. In *Working Notes of EVALITA 2011*. CELCT, Povo, Trento.
- Bosco, Cristina, Simonetta Montemagni, and Maria Simi. 2013. Converting Italian Treebanks: Towards an Italian Stanford Dependency Treebank. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 61–69, Sofia, Bulgaria, August. ACL.
- Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. ACL.
- Chen, Danqi and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

- Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. ACL.
- Cheng, Hao, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional Attention with Agreement for Dependency Parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214, Austin, Texas, November. ACL.
- Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October, 25–29. Association for Computational Linguistics.
- Choi, Jinho D., Joel Tetreault, and Amanda Stent. 2015. It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 387–396, Beijing, China, July. ACL.
- Chu, Y.J. and T.H. Liu. 1965. On the Shortest Arborescence of a Directed Graph. *Science Sinica*, 14:1396–1400.
- Dozat, Timothy and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 2017 International Conference on Learning Representations*, Toulon, France, April 24–26.
- Dozat, Timothy, Peng Qi, and Christopher D. Manning. 2017. Stanford’s Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada, August. ACL.
- Dyer, Chris, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. ACL.
- Edmonds, Jack. 1967. Optimum Branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- Hall, Johan, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939, Prague, Czech Republic, June. ACL.
- Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.
- Kiperwasser, Eliyahu and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.
- Kuncoro, Adhiguna, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753, Austin, Texas, November. ACL.
- Lavelli, Alberto. 2013. An Ensemble Model for the EVALITA 2011 Dependency Parsing Task. In Bernardo Magnini, Francesco Cutugno, Mauro Falcone, and Emanuele Pianta, editors, *Evaluation of Natural Language and Speech Tools for Italian*, pages 30–36, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Lavelli, Alberto. 2014. Comparing State-of-the-art Dependency Parsers for the EVALITA 2014 Dependency Parsing Task. In *Proceedings of the Fourth International Workshop EVALITA 2014*, pages 15–20, Pisa, Italy, December.
- Lavelli, Alberto. 2016. Comparing State-of-the-art Dependency Parsers on the Italian Stanford Dependency Treebank. In *Proceedings of the Third Italian Conference on Computational Linguistics (CLiC-it 2016)*, pages 173–178, Napoli, Italy, December.
- Mazzei, Alessandro. 2015. Simple Voting Algorithms for Italian Parsing. In Roberto Basili, Cristina Bosco, Rodolfo Delmonte, Alessandro Moschitti, and Maria Simi, editors, *Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project*. Springer International Publishing, Cham, pages 161–171.
- McDonald, Ryan, Kobayashi Crammer, and Fernando Pereira. 2006. Spanning Tree Methods for Discriminative Training of Dependency Parsers. Technical Report MS-CIS-06-11, University of Pennsylvania Department of Computer and Information Science, January.

- McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. ACL.
- Montemagni, Simonetta, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*. Springer Netherlands, Dordrecht, pages 189–210.
- Nguyen, Dat Quoc, Mark Dras, and Mark Johnson. 2017. A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 134–142, Vancouver, Canada, August. ACL.
- Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, May.
- Nivre, Joakim and Ryan McDonald. 2008. Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proceedings of ACL-HLT 2008*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.
- Reimers, Nils and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark, September. ACL.
- Sagae, Kenji and Alon Lavie. 2006. Parser Combination by Reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York, New York, June. ACL.
- Sanguinetti, Manuela, Cristina Bosco, Alberto Lavelli, Alessandro Mazzei, Oronzo Antonelli, and Fabio Tamburini. 2018. PoSTWITA-UD: an Italian Twitter Treebank in Universal Dependencies. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May, 7–12. European Language Resources Association (ELRA).
- Shi, Tianze, Liang Huang, and Lillian Lee. 2017. Fast(er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark, September. ACL.
- Shi, Tianze, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017. Combining Global Models for Parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada, August. ACL.
- Surdeanu, Mihai and Christopher D. Manning. 2010. Ensemble Models for Dependency Parsing: Cheap and Good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652, Los Angeles, California, June. ACL.
- Zeman, Daniel and Zdeněk Žabokrtský. 2005. Improving Parsing Accuracy by Combining Diverse Dependency Parsers. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 171–178, Vancouver, British Columbia, October. ACL.

Negated Adjectives and Antonyms in Distributional Semantics: *not similar?*

Laura Aina* **

Universitat Pompeu Fabra, Spain

Raffaella Bernardi†

University of Trento, Italy

Raquel Fernández‡

University of Amsterdam,

The Netherlands

We investigate the relation between negated adjectives and antonyms pairs in English (e.g., not cold vs. hot - cold) using Distributional Semantics. We build vector representations of a set of antonyms and their negations on the basis of their contexts of use, and compare the similarities of the negated adjectives to each of the adjective in their antonym pair. We find that in a distributional semantic model a negated adjective (e.g., not cold) is typically more similar to the adjective itself (cold) than to its antonym (hot). The effect is less strong for antonyms that share their lexical root (morphological; e.g., happy - unhappy). No difference is observed between simple and double negations (e.g., not happy, not unhappy), and contrary and contradictory antonyms (e.g., hot - cold, dead - alive). Our results provides insights on negated adjectives, and in general the type of similarity captured by Distributional Semantics.

1. Introduction

Negation has long posed challenges to researchers in Theoretical and Computational Linguistics (see Horn (1989) and Morante and Sporleder (2012) for overviews). Similarly to logical negation ($\neg p$ is true $\leftrightarrow p$ is false), natural language negation allows to express the falsity of a content. However, when interacting with morphosyntax, semantics and pragmatics, it exhibits a diversity of functions and forms, which go beyond the simplicity of the logical connective (Horn and Kato 2000). We here focus on the **negation of adjectives in English**, as expressed by the particle *not* modifying an adjective – e.g., *not cold*. In particular, we study the relation between these expressions and the antonym pair constituted by the adjective that is negated and its opposite (e.g., *not cold* vs. *cold-hot*). We carry out our investigation using the methods of **Distributional Semantics**, hence representing expressions on the basis of their use in a corpus (Lenci 2008).

It has been noted that when an adjective is negated, addressees not only conclude that the property denoted by it does not apply, but also tend to infer that an alternative property from the same domain applies (e.g., *not cold* \approx *lukewarm*, *hot*, etc.; Horn (1989), Fraenkel and Schul (2008)). For instance, one can take the negation of an adjective to directly convey the opposite of the adjective, that is its antonym (e.g., *not cold* =

* Department of Translation and Language Sciences; Email: laura.aina@upf.edu.

** Part of the work presented in this paper was carried out while at the University of Amsterdam.

† CiMeC, Center for Mind/Brain Sciences and DISI, Information Engineering and Computer Science; Email: raffaella.bernardi@unitn.it

‡ ILLC, Institute for Logic, Language and Computation; Email: raquel.fernandez@uva.nl

hot). However, when a speaker opts for a complex expression when a simpler one is available – choosing to use a negation instead of an antonym – this is typically to serve a particular purpose (Horn 1984). For instance, a range of studies support what is known as *mitigation hypothesis* (see Jespersen (1965) and Horn (Horn 1972) for an early formulation, and Giora (2006) for an overview): a negated adjective tends to be understood as conveying an intermediate meaning between the adjective and its antonym (e.g., *not large* \approx *medium-sized*).

In this work, we study antonymic adjectives and their negations in a distributional semantic model. To this end, we employ an existing dataset of antonyms, whose annotation we further extend; we then obtain distributional representations of these and their negated version. This allows us to conduct a data-driven study of negation and antonymy that covers a large set of instances. We compare pairs of antonyms with distinct lexical roots and those derived by affixation, i.e., **lexical and morphological antonyms** (e.g., *small* - *large* and *happy* - *unhappy*, respectively; Joshi (2012)). Moreover, we investigate the distinction between lexical antonyms that are **contrary or contradictory**, that is, those that do or do not allow an available intermediate value (Fraenkel and Schul 2008): e.g., something *not cold* is not necessarily *hot* - it could be *lukewarm* - but something *not present* is *absent*. As for negations of morphological antonyms, we compare instances of **simple and double negation**, where the latter occurs if the antonym that is negated is an affixal negation (e.g., *not unhappy*).

Our analyses show that, when considering distributional information, a negated adjective (e.g., *not cold*) is typically more similar to the adjective itself (*cold*) than to its antonym (*hot*). Such effect is less strong for antonyms derived by affixation and which then share the same lexical root (e.g., *happy* - *unhappy*). This suggests that there is a tendency for expressions sharing a lexeme to appear in similar contexts. No difference is observed between simple and double negations (e.g., *not happy*, *not unhappy*), and contrary and contradictory antonyms (e.g., *hot* - *cold*, *dead* - *alive*). The latter result contrasts with previous experimental evidence showing that inferential relations between expressions (e.g., *not dead* entails *alive*) affects how similar the negation is perceived to the antonym (Fraenkel and Schul 2008). We hypothesize that this is because distributional similarity is not tailored to capture inferential relations but rather differences in use between expressions. Therefore, even when these seem logically equivalent, they may still emerge as different to a distributional semantic model. Our results provide novel insights on negated adjectives, and in general the type of similarity captured by Distributional Semantics.

2. Related Work: Negation in Distributional Semantics

In Distributional Semantics (henceforth, DS), the distribution of an expression across context in a corpus is taken to be representative of its content, and summarized into a vectorial representation (Lenci 2008; Erk 2012; Turney and Pantel 2010; Baroni, Dinu, and Kruszewski 2014). A vast body of research in Computational Linguistics and Cognitive Science has shown that this methodology is very successful at modeling the meaning of content words (e.g., adjectives). However, these data-driven and bottom-up techniques are not as successful when modeling function words and the complex phenomena that they involve – for instance *not* and in general negation (Bernardi 2014; Boleda and Herbelot 2016). In the case of negation, this is primarily due to the fact that it acts as a truth-reversing operator, whereas in the first place DS lacks a built-in method to account for truth conditions. For this reason, some methods, like that by Garrette et al. (2014), consist of hybrid approaches between distributional and formal semantics,

while others aimed to find a counterpart of logical operations in the distributional space (Widdows and Peters 2003; Coecke, Sadrzadeh, and Clark 2010).

Some approaches have been proposed to specifically model or evaluate the negation of adjectives. Hermann et al. (2013) design a framework where domain and value features of an adjective are separately represented, and when *not* is applied to an adjective, the resulting phrase remains close to others from the same domain of the adjective but its value within the domain changes. Similarly, Rimell et al. (2017) introduce a neural network architecture to learn a mapping from an adjective to the negated version conditioned on the domain of the former. To train this model, they learn negation as a mapping from an adjective to its antonym; a similar idea was also employed by The Pham et al. (2015). As these approaches rest on the assumption that antonyms and negations are equivalent, they do not take into account mitigation effects, nor in general peculiarities of negation that makes it differ from antonymy. However, Socher et al. (2012, 2013) showed that a neural network learning general compositional operations as a byproduct of a sentiment analysis task can actually capture such effects, and correctly taking them into account when assigning fine-grained labels.

Finally, we mention an approach to negation which focuses on noun phrases, but is nevertheless very relevant to our study. Kruszewski et al. (2017) proposed to use similarity relations between expressions as captured by DS to account for the alternatives triggered by the use of a negation. They show that these provide an excellent fit to data of alternative plausibility ratings. For example, in the sentence *This is not a dog, it is a cat* the distributional similarity between *dog* and *cat* is used to measure how plausible the latter is as an alternative to the former. Crucially, this approach showed that, in spite of the difficulties in modeling truth-related aspects of negation, DS can still provide valid contributions to its study. We build on this research line and employ DS as an investigative tool to study the relation between negation and antonymy.

3. Motivation and Data

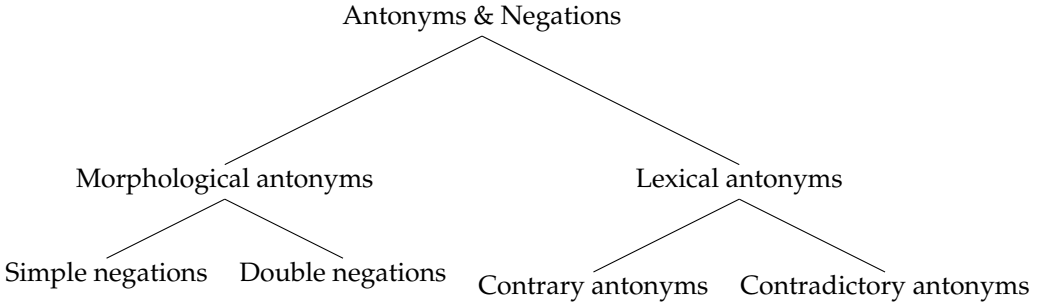
We are interested in how negation acts with respect to pairs of adjectives connected by the lexical relation of **antonymy** (Murphy 2003), that is, associated with opposite properties within the same domain (e.g., *hot* - *cold* with respect to temperature). In particular, we want to compare the negation of one of the antonymic adjectives with itself and its antonym respectively (e.g., *not cold* vs. *cold* and vs. *hot*). Our data of interest are then triples obtained starting from an antonymic pair and negating one of the two items, as in Ex. (1). In the following, we present how we construct such a dataset. Since we are interested in analyzing different types of antonyms and negations, we apply a classification to these data (Figure 1), which we describe and motivate.

Example 1

- (a) $\langle \text{hot}, \text{cold}, \text{not } \{\text{hot} \parallel \text{cold}\} \rangle$
- (b) $\langle \text{happy}, \text{unhappy}, \text{not } \{\text{happy} \parallel \text{unhappy}\} \rangle$

3.1 Dataset

In order to build a dataset of triples as in Ex. 1, it is sufficient for us to have a dataset of antonymic pairs of adjectives. Indeed, we can automatically obtain the negation of each adjective by simply making it precede by *not*. Note that for each pair we can obtain two triples, by negating each of the adjectives in turn.

**Figure 1**

Categorisation of the triples consisting of an antonymic pair and a negated adjective employed in our experiments.

As a dataset of antonyms, we make use of a subset of the **Lexical Negation Dictionary** by Van Son et al. (2016). This consists of word pairs tagged as antonyms in WordNet (Fellbaum 1998) classified into different types of lexical negation, following the categorization of Joshi (2012). In particular, lexical negation is taken to include both *affixal negations* (e.g., *perfect* - *imperfect*), and *regular antonyms* (e.g., *hot* - *cold*). The former is in turn split into *direct* and *indirect* negation, depending on whether the meaning of the affixed word actually expresses an opposite property of the non-affixed counterpart (e.g., *imperfect* expresses a different degree than *perfect* in the scale of perfection, while the difference between *famous* - *infamous* is not about the degree of fame and they are not incompatible with each other). Therefore, we consider as antonymic adjectives only those pairs in the Lexical Negation dictionary that either involve a direct affixal negation on one side, and a relation of regular antonymy on the other. We refer to these as **lexical antonyms**, i.e., with distinct lexical roots (e.g., *cold* - *hot*), or **morphological antonyms**, i.e., sharing the lexical root (e.g., *happy* - *unhappy*). The motivation to use this dataset is that it leverages a large-scale resource such as WordNet, but also allows us to filter out pairs that do not actually correspond to the relation of antonymy in the sense of direct opposition. Finally, as it comes equipped with the classification between lexical and morphological antonyms, it enables us to investigate differences between these two groups (see following section).

From this list of antonyms, we build our dataset of triples as explained above. Our analyses methods leverage on occurrences of expressions in a corpus, which we use to obtain their distributional representations (see details in Section 4). To ensure that we only consider representations based on a relatively high number of occurrences, we enforce a frequency threshold: to be employed in the analyses, each of the elements of the triple (the two adjectives and the negation) needs to occur at least 100 times in the corpus. Table 1 shows the final number of triples for each class and the average frequency of their elements after this filtering. As it can be expected, negated adjectives are overall less frequent than adjectives.¹

¹ Full list of triples at <https://lauraina.github.io/data/notadj.csv>

**Figure 2**

Interaction of negation with an antonym pair: negation shifts the meaning of an adjective towards its antonym.

3.2 Antonyms: Lexical vs. Morphological

We are interested in investigating the relationship between negation by *not* and antonymy. A non-parsimonious expression – e.g., a negated adjective – tends to trigger the implicature that a different meaning from a simpler alternative – an antonym – is intended (Grice 1975; Horn 1984). For instance, it has been shown that one of the functions of negation is to act as a modifier of degree (Giora et al. 2005): it alters the meaning of the adjective it applies to and shifts it more or less close to its antonym (Figure 2). Such mitigation in meaning has been explained in pragmatic terms, but also as a result of the representational process: it arises due to the interaction between the negativity of the particle *not* and the meaning of the adjective, which is retained as accessible in memory (Giora et al. 2005).

We aim to investigate the semantic shift that results from applying negation to an adjective, and the extent that this makes it closer to the antonym. We analyze this using the measures of similarity yielded by a distributional semantic model – e.g., is *not hot* closer to *cold* than to *hot*? Differently from previous studies, we here compare negated adjectives and antonyms solely on the basis of their contexts of use. Moreover, we compare the behavior of lexical and morphological antonyms in this respect. As we mentioned earlier, these two classes of antonyms are usually taken to express the same lexical relation, namely opposition – and to be different only on morphological terms. However, such difference might affect their relation with negated adjectives: indeed, affixal negations have a morphological structure that resembles negated adjectives (e.g., *un-happy* vs. *not happy*). It may then be that antonyms with a negative affix are more similar to the negated adjective than antonyms with a distinct lexical root – e.g., is *not frequent* closer to *infrequent* than *not hot* to *cold*? To investigate this, we compare triples derived from lexical and morphological antonyms.

3.3 Lexical Antonyms: Contrary vs. Contradictory

An important distinction within the lexical antonyms group is that between contradictory and contrary pairs (Clark 1974). In the case of contradictory antonyms, the negation of one of the adjectives entails the truth of the other, without the availability of a mid-value (e.g., *not dead* implies *alive*). The opposite is true for contrary pairs (e.g., *not hot* does not imply *cold*, since a mid-value, such as *lukewarm*, exists). In other terms, contradictory pairs constitute a dichotomy, while contrary ones lie in a continuum. Fraenkel and Shul (2008) provided psycholinguistic results showing that if an adjective is part of a contradictory pair, its negation is interpreted as being closer to the related antonym than if it is instead part of a contrary pair (e.g., *not dead* is interpreted as being closer to *alive* than *not small* to *large*). This corresponds to the intuition that when

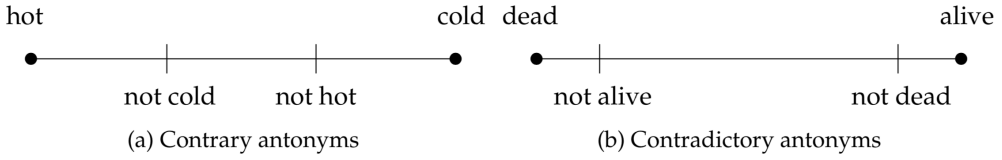


Figure 3

Example of the effects observed by Fraenkel and Shul (2008): a negated adjective that is part of a contradictory pair typically expresses a meaning that is closer to the adjective’s antonym than a negated adjective that is part of a contrary pair.

negation shifts the meaning of the adjective towards its opposite, such shift is bigger if a mid-value between these is not available (see Figure 3). However, Fraenkel and Shul (2008) also noted that, in spite of this general result, some variation may occur: even for contradictory pairs it may be possible to conceive a context where a mid-value interpretation is available (e.g., *not dead* \approx *half-dead*; Paradis and Willners (2006)).

We replicate the analysis of Fraenkel and Shul (2008) with DS, where similarities between expressions can be quantified in terms of the geometric distance between their distributional representations (details in Section 4). The antonyms pairs in the Lexical Negation Dictionary do not come with a classification into contrary and contradictory; in general, to the best of our knowledge, no large-scale dataset annotated with this information is available. Therefore, the three authors independently annotated the lexical antonym pairs extracted by the Lexical Negation Dictionary as contrary or contradictory, following the definitions reported by Fraenkel and Shul (2008).² In particular, we tag as contrary those pairs a, b such that it is acceptable to say that something is “neither a nor b ”, and viceversa for contradictory (“neither *hot* nor *cold*” vs. “neither *dead* nor *alive*”). Antonyms could also be tagged as *unclear*, if none of these options clearly fit.³ Not surprisingly, the inter-annotator agreement is only moderate (Fleiss’ $k = 0.37$). As we mentioned above, the distinction into contrary and contradictory antonyms presents some limitations: these are likely to cause difficulties when scaling the classification up to a large set of antonyms and attempting to reach agreement over it. Therefore, our low agreement in the annotation further underscores the possibility that the availability of an alternative between two antonyms may be a contextual matter, and that the contrary vs. contradictory distinction may rather be a graded one. We leave this aspect to be clarified by future research and, for the purpose of our analysis, only consider antonyms pairs classified with full agreement. Table 1 reports the values after such filtering: as it can be seen, we had to exclude almost 50% of the lexical antonyms triples. In particular, this filtering leaves us with a small number of triples involving contradictory antonyms.

3.4 Morphological Antonyms: Simple vs. Double Negations

In the case of morphological antonyms, one of the two adjectives is an affixal negation, and hence already contains a negating prefix (such as *im-* in *imperfect* or *un-* in *unhappy*):

² We only annotated lexical antonyms pairs that we could analyze in our setup. That is, we first filtered all triples on the basis of their frequency of the corpus employed (see Section 4) and then annotated the lexical antonyms in these data.

³ Annotation guidelines at <https://lauraina.github.io/data/notadj.pdf>.

Table 1

Total number of triples $\langle a_1, a_2, \text{not } \{a_1 \| a_2\} \rangle$ used in the experiment, after filtering by frequency and annotation agreement, and average frequency of adjectives and negated adjectives in these triples per class.

	# triples	frequency	
		adj.	not adj.
Lexical antonyms	198	254K	1K
– contrary	68	337K	1K
– contradictory	28	298K	1K
Morphological antonyms	185	83K	1.8K
– simple negations	157	85K	2K
– double negations	28	122K	0.9K

adding *not* thus gives rise to a double negation (e.g., *not imperfect*; see Figure 2). Since our dataset of triples is obtained by negating both of the adjectives in the set of antonymic pairs, datapoints involving morphological antonyms encompass both simple and double negations (see Ex. 1). Double negations have been widely studied in the literature due to their difference with double negation in logic (e.g., Bolinger (1972), Krifka (2007), Tessler and Franke (2018)). While in logic two negations cancel each other out ($\neg\neg p \equiv p$), in natural language double negations are typically employed to weaken the meaning of the adjective that is negated twice (e.g., *not unhappy* \neq *happy*). We here test whether evidence for this effect is found in a distributional semantic model: in particular, if two negations were equivalent to no negation at all we would expect that the negation of an affixal negation (e.g., *not unhappy*) is particularly close to the antonym (e.g., *happy*). Therefore, we check whether simple (e.g., *not happy*) and double (e.g., *not unhappy*) negations exhibit similar trends in relation to an antonym pair (*happy* vs. *unhappy*).

We classify our morphological antonyms data into simple and double negations, as follows. From the Lexical Negation Dictionary, we know which adjective among an antonym pair is the affixal negation; if a triple includes a negation of an affixal negation (e.g., *not imperfect*), it is classified into the double negations groups, and viceversa for simple negations. As Table 1 shows, we could only consider a small number of double negations; this is due to the fact that these expressions are rarely produced, and therefore few occur in our corpus more than the frequency threshold.

4. Methods

Previous studies about negation described its effect on an adjective as a meaning shift towards the antonym, that can be measured in terms of **semantic similarity** (Fraenkel and Schul 2008). DS offers us a data-driven method of quantifying this, leveraging the occurrences of antonyms and negations in a corpus. Within this framework, expressions - here, adjectives and their negated versions - are represented as vectors of continuous values, summarizing their distribution across contexts of use in a corpus. Crucially, we can interpret the proximity relations of the expressions in the resulting high-dimensional space (e.g., cosine between their vectors) in terms of similarity relations between them. By definition, this measures similarity of distribution: the more two expressions appear in the same contexts, the more they will be distributionally similar.

While representing words in this fashion is standard for adjectives and in general word units, it is more atypical as a way to represent multi-word phrases, such as a negations. These are more typically represented using compositional operations (Baroni

2013). In the case of negation, these often incorporate assumptions about its resulting behavior (e.g., Hermann et al. (2013), Rimell et al. (2017)). To avoid introducing any bias, we simply represent negated adjectives by treating them as a single unit (see details below). This approach allows us to study negation in a bottom-up fashion, moreover covering a large set of instances. An important caveat is in place: as in standard Distributional Semantics, each expression is assigned one vector abstracting away from all their uses. Therefore, we focus on the main regularities in the use of adjectives and negated adjectives – as captured in their distributional vectors – leaving an investigation of their context-sensitive behavior to future work.

4.1 Distributional Semantic Model

To build a distributional semantic model with negated adjectives, we employ an existing algorithm but apply a particular pre-processing of the training corpus. In particular, we want the vocabulary of our model to include, besides word units, also negated adjectives. We pre-process the training corpus as follows: adjacent occurrences of the particle *not* and an adjective are merged (e.g., *not cold* \leadsto *not_cold*), therefore treating each negated adjective as a single type, independent of the related adjective. Moreover, we employ part of speech labels for adjectives, to distinguish occurrences of a form as a different part of speech (e.g., *poor* as adjective or noun, as in *the poor*). Finally, we remove function words, as to avoid syntactic differences between adjectives and negated adjectives (e.g., the former ones appear both in predicative and attributive position, while the latter ones almost exclusively in attributive position).

The corpus we employ is the concatenation of UkWaC and Wackypedia-En corpora (2.7B tokens; Baroni et al., (2009)). We train a word2vec CBOW model (Mikolov et al. 2013) on this corpus, setting its hyperparameters as in the best performing model by Baroni et al. (2014).⁴ We are interested in investigating characteristics of antonyms and negated adjectives in a distributional semantic model that is not fine-tuned to a particular task and where no assumptions about the structure of its space are incorporated. Therefore, we do not carry out any hyperparameters search, nor we employ any ad hoc techniques aimed at, for example, amplifying the distances between antonyms in the semantic space (such as those by Nguyen et al. (2016) or The Pham et al. (2015)). We assess the quality of the induced model through a similarity relatedness task, where we find that it achieves satisfying performances.⁵

4.2 Quantitative Analysis

We consider triples as in Ex. 1, derived as described in Section 3. Given a triple $\langle a_i, a_j, \text{not } a_i \rangle$ (e.g., *cold, hot, not cold*), we define the following score:

$$\text{Shift} := \text{Sim}(\text{not } a_i, a_j) - \text{Sim}(\text{not } a_i, a_i) \quad (1)$$

⁴ Vectors size: 400; window size: 5; minimum frequency: 20; sample: 0.005; negative samples: 1. We employ the Gensim implementation of CBOW from <https://radimrehurek.com/gensim/models/word2vec>

⁵ Spearman's ρ of 0.75 on the MEN dataset (Bruni, Tran, and Baroni 2014); see results by Baroni et al. (2009) for a comparison.

Table 2

Average *Shift* scores, with standard deviation, for each category. ***: significant difference between categories in the row ($p < 0.001$, Welch's *t*-test).

Lexical antonyms	-.19 ($\pm .16$)	Morphological antonyms	-.04 ($\pm .16$)	***
Contrary antonyms	-.18 ($\pm .15$)	Contradictory antonyms	-.19 ($\pm .16$)	
Simple negations	-.03 ($\pm .17$)	Double negations	-.06 ($\pm .11$)	

where $i \neq j$, and $\text{Sim}(\text{not } a_i, a_j)$ and $\text{Sim}(\text{not } a_i, a_i)$ are the cosine similarities of the negated adjective with the antonym and the adjective, respectively. *Shift* measures how much closer a negated adjective is to the antonym than to the adjective it self (i.e., how much closer *not cold* is to *hot* than to *cold*), and hence how much negation shifts the meaning of an adjective towards that of the antonym. When the *Shift* score positive, the negated adjective is closer to the antonym, and viceversa. Due to the well-known tendency of antonym pairs to be close in a distributional space (Mohammad et al. 2013), the absolute value of *Shift* is not expected to be high: since antonyms tend to be close to each other, a vector that is close to one is also likely to be close to the other. However, this is practically not a problem, as by comparing the similarities with the two antonyms, respectively, we can still assess whether a higher proximity is registered towards one of them. For instance, Kim and De Marneffe (2013) have shown that, in spite of the relative proximity of two antonyms, meaningful relationships among other members of their domain can still be captured: they were able to retrieve adjectival scales by looking at intermediate points between the two antonyms' vectors.

5. Results

Table 2 shows the scores across the different categories described in Section 3. Example triples for each category are given in Table 3, together with the nearest adjectives of each element in the triple. Figure 4 offers a visualization of the results for the different categories.⁶

5.1 Lexical vs. Morphological Antonyms

The average *Shift* scores of both classes are negative, showing that a negated adjective is typically closer to the adjective than to the antonym. Indeed, as shown in Table 3, the nearest neighbor of a negated adjective is often the related adjective.⁷ At first glance, one could interpret this result as supporting the idea that negated adjectives express an intermediate meaning between that of the adjective and the antonym (e.g., *not small* is close to *normal-sized*). More in general, considering the setup of our experiments, it shows that negated adjectives have a profile of use that is more similar to that of the adjective than to the antonym.

⁶ The Figure serves as a visualization of the results for each category, and not for the particular triple that is reported as example. That is, we locate the elements of the example triples on the basis of the average *Shift* score of their category, and not the score of the specific triple.

⁷ Note that we treated negated adjectives and adjectives as completely separate types. In particular, the occurrence of a negated adjective does not count also as an occurrence of an adjective. Therefore, the result cannot be led back to introducing an overlap in distribution.

Table 3
Nearest adjectives in the semantic space for the three elements in some sample triples.

Contrary antonyms	small: <i>large, tiny, smallish, sizeable, largish</i>	large: <i>small, sizeable, huge, vast, smallish</i>	not small: <i>small, smallish, normal-sized, largish, middle-sized</i>
Contradictory antonyms	dead: <i>drowned, lifeless, half-dead, wounded, alive</i>	alive: <i>dead, awake, unharmed, beloved, tortured</i>	not dead: <i>dead, half-dead, alive, comatose, lifeless</i>
Simple negations	similar: <i>analogous, identical, comparable, dissimilar, same</i>	dissimilar: <i>similar, different, distinct, unrelated, identical</i>	not similar: <i>similar, dissimilar, identical, distinguishable, analogous</i>
Double negations	happy: <i>glad, pleased, contented, nice, kind</i>	unhappy: <i>disappointed, dissatisfied, unsatisfied, resentful, anxious</i>	not unhappy: <i>unhappy, adamant, disappointed, dismayed, unimpressed</i>

The two classes of antonyms differ significantly in the extent of this effect: the Shift score is higher for morphological antonyms than for lexical ones. We found that this is due to the fact that negated adjectives and morphological antonyms are significantly closer to each other than it is the case for lexical antonyms, whereas there is not a significant difference between how close the negation is to the adjective.⁸ The higher similarity between negations and morphological antonyms can be justified by the fact that one of the two morphological antonyms is an affixal negation. Its structure would then be much more similar to that of negated adjectives than it is the case for lexical antonyms: both affixal negations and negations by *not* are formed by a negative particle and the adjective itself (e.g., *not* vs. *un-*, *im-* etc. + adjective). This might impact on the similarity between, i.e., *not happy* and *unhappy*, as well as that between *not unhappy* and *happy* (see Section 5.3 for a comparison between simple and double negations).

Overall, the picture that emerges is one where sharing a lexeme – in particular, as a separate word – impacts on the distributional similarity between expressions. Indeed, negated adjectives tend to be more similar to the original adjective – which they specifically include as a separate word – than to the antonym (e.g., *not cold* - *cold* vs. *hot*). However, if also the antonym shares the lexical root – which is the case for morphological antonyms (e.g., *not imperfect* – *perfect*), this effect is less strong, as the negated adjective is also quite close to the antonym. One way to explain this is by positing that the contexts of use of an expression may be affected by a lexeme that they include, due to the connotations that this carries. For instance, in a context where too-direct expressions are to be avoided for politeness, expressions like *incorrect* or *not correct* – sharing *correct* – may be preferred to *wrong*.

5.2 Contrary vs. Contradictory Antonyms

In contrast to results from the linguistic literature (see Section 3), the behavior of contrary and contradictory antonym pairs is not significantly different in our analysis. When we look into a distributional space, even for contradictory antonyms, the negated

⁸ The average Sim(not a_i , a_j) is 0.43 and 0.16 for morphological and lexical antonyms, respectively

adjectives tend to be more similar to the adjective itself than to the antonym (e.g., *not dead* is closer to *dead* than to *alive*).

While this result may seem counterintuitive at first, we posit the following explanation. The contrary vs. contradictory distinction taps into inferential relations between expressions - e.g., *not present* implies *absent*. These constraint the potential readings of negation (e.g., an intermediate meaning is not available, so it cannot be intended) and affect speakers when prompted to judge the similarity between negations and antonyms (Fraenkel and Schul 2008). However, Distributional Semantics organizes its lexicon in terms of general relatedness relations, with no built-in method to capture inferential relations between expressions (Bernardi 2014; Boleda and Herbelot 2016). Yet, this does not mean that distributional similarity is not informative regarding negation. It indeed captures a different and possibly complementary type of similarity from that tackled, for example, in the experiments of Fraenkel and Shul (2008). Even if the negation of an adjective and the antonym may seem intuitively equivalent, the use of one or the other may serve different functions (e.g., contradicting an expectation, politeness, emphasis etc.) leading them to appear in different contexts. Therefore, the negation of an adjective from a contradictory pair may not be so similar to its antonym when looking at their distribution. One could argue that this effect is a shortcoming of DS; on the contrary, we regard it be an interesting property that naturally arises as an artefact of representing expressions solely on the basis of their contexts of use.

Finally, we find that, since continuous representations are able to capture nuanced differences, the alleged dichotomy between contrary and contradictory antonyms tends to become a continuum in the distributional space. For example, one of the closest adjectives to *not dead* is *half-dead*: this suggests a gradability of the *dead-alive* scale, in spite of the pair being categorized as contradictory. This further underscores the complexity of the contrary and contradictory distinction which we had already encountered in the annotation procedure.

5.3 Simple vs. Double Negations

There is not a significant difference between negated adjectives that are instances of simple and double negations: crucially, it is not the case that double negations are very close to the antonym as a result of the two negations canceling each other out (e.g., *not unhappy* is closer to *unhappy* than to *happy*). The results could be interpreted in terms of mitigation: for instance, *not unhappy* is close to *unimpressed*, a mid-value between *happy* and *unhappy* (Table 3). More in general, following an analogous rationale to what noted earlier, it suggests that the contexts of use of double negations are more similar to the ones of the adjective that is negated than to those of its antonym. Indeed, double negations typically appear in contexts where the use of the “logically” equivalent alternative (i.e., the antonym) is to be avoided for pragmatic reasons, as possibly too strong or direct (e.g., *not unproblematic* vs. *problematic*; Horn, (1984)).

6. Discussion and Conclusion

We have investigated negated adjectives using the tools of DS, which allows us to quantify the similarities between expressions on the basis of how they are used. Our analyses show that, when considering contexts of occurrence, negating an adjective does not make it closer to the antonym than it is to the adjective itself. We hypothesized that this effect may partially be due to mitigation effects (Giora et al. 2005), but more

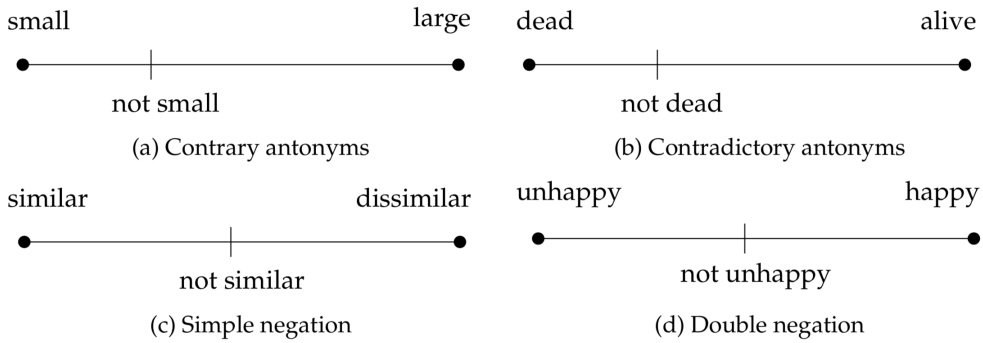


Figure 4

Visualization of the results; the negation is located as distant from each antonym following the average *Shift* score in Table 2 (e.g., center of the scale: *Shift* = 0; closer to the adjective: *Shift* < 0).

in general due to the different functions that these expressions are used in, which ultimately reflect their contexts of use. This follows from the type of methodology which we employ in our study, namely the distributional one.

Our results align with observations in the linguistic literature. Language has been noted to exhibit a force to diversification of meanings, such that full synonymic expressions tend to be avoided (Kiparsky 1982; Clark 1992). In particular, according to the *division of pragmatic labor* posited by Horn (1984), if a speaker opts for a more complex or less fully lexicalized expression over a simpler alternative (e.g. a negated adjective over an antonym) this is justified by a particular function. This could be for example that of expressing an intermediate meaning, but also retaining the emphasis on a rejected concept, or attenuating the strength of a statement. Our results suggest that distributional representations may be sensitive to such differences in use. In particular, we found that sharing a lexical root affects how similar the distributions are: this suggests that certain lexemes are associated to particular contexts. Moreover, we could not conclude that relations of distributional similarity align with inferential relations between expressions, that is *not dead* is not closer to *alive* than to *dead*. This suggests that: 1) again, the lexical root has an impact on the contexts of occurrence; 2) alternative expressions, such as antonyms or negated adjectives, are not fully interchangeable, and therefore used in the same contexts, even when logically equivalent (e.g., the case of contradictory antonyms and double negations). Further research may shed light on which type of contexts characterize the two types of expression, for example through a corpus study. Moreover, it would be interesting to assess which other properties negated adjectives have in a distributional space, such as their interaction with scalar dimensions (e.g., *not hot* vs. *freezing*, *cold*, *lukewarm*, *hot* etc.; Wilkinson and Tim (2016)) and implicatures (Van Tiel et al. 2016).

Our study is in line the proposal put forward by Kruszewski et al. (Kruszewski et al. 2017) that, even though Distributional Semantics may not be the right tool to represent truth-related aspects of meaning, it can be still very useful to study certain aspects of negation. In our case, we have showed that DS is probably not the right tool to capture inferential relations involving negations, but it can be used to quantify how similar the use of negations is to that of other expressions. For the purpose of this study we have focused on one distributional semantic model; however, it would be interesting to test

the robustness of our findings through an evaluation of various models. Moreover, one could apply an analogous methodology to analyze other types expressions which are taken to convey almost identical semantic content but may be used in different ways; for example, the quantifiers *most* and *more than a half* in English (Hackl 2009), or different forms of absolute superlatives in Italian (e.g., *molto bello* - *bellissimo*; Berlanda (2013)).

Our study primarily relates to research questions in Linguistics; however, we regard our results to also be of interest for Natural Language Processing. Aspects of negation as the ones we studied, as well as their effect on distributional semantic models, can be critical for tasks like stance detection or sentiment analysis (e.g., what does it imply that a costumer is *not happy* or *not unhappy* with a product?; Wiegand et al, (2010)).

Acknowledgments

We thank Gemma Boleda and Malvina Nissim for their useful suggestions. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154), and by the Catalan government (SGR 2017 1575). This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.



References

- Baroni, Marco. 2013. Composition in distributional semantics. *Language and Linguistics Compass*, 7(10):511–522.
- Baroni, Marco, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.
- Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 238–247, Baltimore, Maryland, USA, June 22–27.
- Berlanda, Sara. 2013. Constructional intensifying adjectives in Italian. In *Proceedings of the 9th Workshop on Multiword Expressions*, pages 132–137, Atlanta, Georgia, 13–14 June.
- Bernardi, Raffaella. 2014. Distributional semantics: A Montagovian view. In Claudia Casadio, Bob Coecke, Michael Moortgat, and Philip Scott, editors, *Categories and Types in Logic, Language, and Physics*. Springer, pages 63–89.
- Boleda, Gemma and Aurélie Herbelot. 2016. Formal distributional semantics: Introduction to the special issue. *Computational Linguistics*, 42(4):619–635.
- Bolinger, Dwight. 1972. *Degree words*. Walter de Gruyter.
- Bruni, Elia, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49(2014):1–47.
- Clark, Eve V. 1992. Conventionality and contrast: pragmatic principles with lexical consequences. In Eva Kittay and Adrienne Lehrer, editors, *Frames, Fields, and Contrasts: New Essays in Semantic and Lexical Organization*. Routledge, pages 171–188.
- Clark, Herbert H. 1974. Semantics and comprehension. In Thomas A. Sebeok, editor, *Current trends in linguistics: Linguistics and adjacent arts and sciences*, volume 12. Mouton, pages 1291–1428.
- Coecke, Bob, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING): Tutorial Abstracts*, pages 1–4, Beijing, China, August 23–27.
- Erk, Katrin. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.
- Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*. MIT press.
- Fraenkel, Tamar and Yaacov Schul. 2008. The meaning of negated adjectives. *Intercultural Pragmatics*, 5(4):517–540.

- Garrette, Dan, Katrin Erk, and Raymond Mooney. 2014. A formal approach to linking logical form and vector-space lexical semantics. In Harry Bunt, Johan Bos, and Stephen Pulman, editors, *Computing meaning*, volume 4. Springer, pages 27–48.
- Giora, Rachel. 2006. Anything negatives can do affirmatives can do just as well, except for some metaphors. *Journal of Pragmatics*, 38(7):981–1014.
- Giora, Rachel, Noga Balaban, Ofer Fein, and Inbar Alkabetz. 2005. Negation as positivity in disguise. In Albert N. Katz and Herbert L. Colston, editors, *Figurative language comprehension: Social and cultural influences*. Lawrence Erlbaum Associates, pages 233–258.
- Grice, H. Paul. 1975. Logic and conversation. *Syntax and Semantics*, pages 41–58.
- Hackl, Martin. 2009. On the grammar and processing of proportional quantifiers: most versus more than half. *Natural Language Semantics*, 17(1):63–98.
- Hermann, Karl Moritz, Edward Grefenstette, and Phil Blunsom. 2013. “Not not bad” is not “bad”: A distributional account of negation. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 74–82, Sofia, Bulgaria, August, 9.
- Horn, Laurence R. 1972. *On the Semantic Properties of Logical Operators in English*. University of California, Los Angeles.
- Horn, Laurence R. 1984. Toward a new taxonomy for pragmatic inference: Q-based and R-based implicature. *Meaning, form, and use in context: Linguistic applications*, pages 11–42.
- Horn, Laurence R. 1989. *A natural history of negation*. University of Chicago Press.
- Horn, Laurence R. and Yasuhiko Kato. 2000. Introduction: Negation and polarity at the millennium. In Laurence R. Horn and Yasuhiko Kato, editors, *Negation and Polarity. Syntactic and Semantic Perspectives*. Oxford University Press, pages 1–19.
- Jespersen, Otto. 1965. *The philosophy of grammar*. University of Chicago Press.
- Joshi, Shrikant. 2012. Affixal negation: direct, indirect and their subtypes. *Syntaxe et sémantique*, 13(1):49–63.
- Kim, Joo-Kyung and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1625–1630, Seattle, Washington, USA, October, 18–21.
- Kiparsky, Paul. 1982. Word-formation and the lexicon. In *Proceedings of the Mid-America Linguistics Conference*.
- Krifka, Manfred. 2007. Negated antonyms: Creating and filling the gap. In Uli Sauerland and Penka Stateva, editors, *Presupposition and implicature in compositional semantics*. Palgrave MacMillan, pages 163–177.
- Kruszewski, Germán, Denis Paperno, Raffaella Bernardi, and Marco Baroni. 2017. There is no logical negation here, but there are alternatives: Modeling conversational negation with distributional semantics. *Computational Linguistics*, 42(4).
- Lenci, Alessandro. 2008. Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, 20(1):1–31.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of 2013 International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA, May, 2–4.
- Mohammad, Saif M., Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2013. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.
- Morante, Roser and Caroline Sporleder. 2012. Modality and negation: An introduction to the special issue. *Computational Linguistics*, 38(2):223–260.
- Murphy, Lynne. 2003. *Semantic relations and the lexicon: Antonymy, synonymy and other paradigms*. Cambridge University Press.
- Nguyen, Kim Anh, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany, August 7–12.
- Paradis, Carita and Caroline Willners. 2006. Antonymy and negation—the boundedness hypothesis. *Journal of pragmatics*, 38(7):1051–1080.
- Rimell, Laura, Amandla Mabona, Luana Bulat, and Douwe Kiela. 2017. Learning to negate adjectives with bilinear models. In *Proceedings of the 15th Annual Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 71–78, Valencia, Spain, April 3–7.
- Socher, Richard, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 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 (EMNLP-CoNLL), pages 1201–1211, Jeju Island, Korea, July, 12–14.
- Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D Manning, Andrew Y. Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, Seattle, USA, October, 18–21.
- Tessler, Michael Henry and Michael Franke. 2018. Not unreasonable: Carving vague dimensions with contraries and contradictions. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, Madison, Wisconsin, USA, July, 25–28.
- The Pham, Nghia, Angeliki Lazaridou, and Marco Baroni. 2015. A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 21–26, Beijing, China, July 26–31.
- Turney, Peter D. and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.
- van Son, Chantal, Emiel van Miltenburg, and Roser Morante Vallejo. 2016. Building a dictionary of affixal negations. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, Osaka, Japan, December, 12.
- Van Tiel, Bob, Emiel Van Miltenburg, Natalia Zevakhina, and Bart Geurts. 2016. Scalar diversity. *Journal of Semantics*, 33(1):137–175.
- Widdows, Dominic and Stanley Peters. 2003. Word vectors and quantum logic: Experiments with negation and disjunction. *Mathematics of language*, 8(141–154).
- Wiegand, Michael, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing (NeSP-NLP)*, pages 60–68, Uppsala, Sweden, July 10.
- Wilkinson, Bryan and Oates Tim. 2016. A gold standard for scalar adjectives. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia, May, 23–28.

Event Knowledge in Compositional Distributional Semantics

Ludovica Pannitto*
Università di Pisa
Università di Trento

Alessandro Lenci**
Università di Pisa

The great majority of compositional models in distributional semantics present methods to compose vectors or tensors in a representation of the sentence. Here we propose to enrich one of the best performing methods (vector addition, which we take as a baseline) with distributional knowledge about events. The resulting model is able to outperform our baseline.

1. Compositional Distributional Semantics: Beyond vector addition

Linguistic competence entails the ability to understand and produce an unbounded number of novel, complex linguistic expressions. The comprehension of such expressions involves the construction of a semantic representation that, following a common statement for the so-called *principle of compositionality*, is said to be a function of the meaning of its parts and their syntactic modes of combination (Partee 1984).

These representations are needed to support human reasoning about the event or situation that is cued by language use. Consider for instance the different implications of sentences 1 and 2:

- (1) After the landing, the pilot switched off the engine.
- (2) After the rally, the pilot switched off the engine.

While the two sentences share the proposition *the pilot switched off the engine*, we are likely to infer different things, for instance, about the *engine* that is being switched-off (i.e., the fact that in (1) it refers to an airplane or a ship while in (2) it refers to a car). Other aspects are involved as well: different inferences could be made upon which other participants are expected to perform further actions, for example *cabin crew*, *control tower*, *passengers* might be involved in the first scenario, but are definitely cut out from the second. Words like *landing* and *rally* cue in fact very different situations in the two sentences, creating different sets of expectations about the described event.

We expect our computational resources to be able to model such phenomena, that make up the very core of language use. In the last decades, distributional semantics has provided a solid framework for the representation of word meaning (Lenci 2018), and various approaches have been also introduced in order to extend vector models of meaning beyond the word level, for representing the meaning of more complex structures such as sentences. Compositional distributional semantics has mainly adopted

* CIMEC - Corso Bettini 31, 38068 Rovereto (TN), Italy. E-mail: ludovica.pannitto@unitn.it
** Dipartimento di Filologia, Letteratura e Linguistica - 56126 Pisa (PI), Italy.
E-mail: alessandro.lenci@unipi.it

a *syntactically transparent* model of semantic composition (Jackendoff 1997), and has addressed the problem of compositionality mainly relying on this standard, Fregean approach, namely considering the lexicon as a pretty much fixed set of word-meaning pairs, and representing sentence meaning as the algebraic composition of pre-computed semantic representations. Composing word representations into larger phrases and sentences notoriously represents a big challenge for distributional semantics (Lenci 2018). Various approaches have been proposed ranging from simple arithmetic operations on word vectors (Mitchell and Lapata 2008), to algebraic compositional functions on higher-order objects (Baroni, Bernardi, and Zamparelli 2014; Coecke, Clark, and Sadrzadeh 2010), as well as neural networks approaches that build so-called sentence embeddings (Kiros et al. 2015; Conneau et al. 2017).

Among all proposed compositional functions, vector addition still shows remarkable performances on various tasks, such as phrase similarity or paraphrase detection (Asher et al. 2016; Blacoe and Lapata 2012; Rimell et al. 2016), beating more complex methods, such as the Lexical Functional Model (Baroni, Bernardi, and Zamparelli 2014). However, the success of vector addition is quite puzzling from the linguistic and cognitive point of view: the meaning of a complex expression is not simply the sum of the meaning of its parts, and the contribution of a lexical item might be different depending on its syntactic as well as pragmatic context.

The majority of available models in literature assumes the meaning of complex expressions like sentences to be a vector (i.e., an embedding) projected from the vectors representing the content of its lexical parts. However, as pointed out by Erk and Padó (2008), while vectors serve well the cause of capturing the semantic relatedness among lexemes, this might not be the best choice for more complex linguistic expressions, because of the limited and fixed amount of information that can be encoded. Moreover events and situations, expressed through sentences, are by definition inherently complex and structured semantic objects. Actually, assuming the equation “meaning is vector” is eventually too limited even at the lexical level.

On the other hand, factors that have been long assumed to lie outside the lexicon, such as pragmatic or world knowledge, have proven to be processed together with lexical knowledge, playing a significant role in comprehension very early in processing, guiding the hearer’s expectations about the upcoming input. Psycholinguistic evidence shows that lexical items activate a great amount of generalized event knowledge (GEK) (Elman 2011; Hagoort and van Berkum 2007; Hare et al. 2009), and that this knowledge is crucially exploited during online language processing, constraining the hearer’s expectations about upcoming linguistic input (McRae and Matsuki 2009). GEK is concerned with the idea that the lexicon is not organized as a dictionary, but rather as a network, where words trigger expectations about the upcoming input, influenced by pragmatic knowledge along with lexical knowledge. Therefore sentence comprehension can be phrased as the identification of the event that best explains the linguistic cues used in the input (Kuperberg and Jaeger 2016).

Here, we introduce **MEDEA** (Merging Event knowledge and Distributional vEctor Addition), a structured compositional distributional model of sentence meaning which integrates vector addition with generalized event knowledge activated by lexical items (Section 2). MEDEA is directly inspired by the model in Chersoni, Lenci, and Blache (2017) and relies on two major assumptions:

- lexical items are represented with embeddings within a network of syntagmatic relations encoding prototypical knowledge about events;
- the semantic representation of a sentence is a structured object incrementally integrating the semantic information cued by lexical items.

Our aim is to integrate the evidence on the role played by event knowledge during language processing in a linguistically motivated model for compositional semantic representations.

We test MEDEA (Section 3) on two datasets for compositional distributional semantics in which addition has proven to be hard to beat: the first is RELPRON (Rimell et al. 2016), a popular dataset for the similarity estimation between compositional distributional representations; the second is the transitive sentences similarity dataset (Kartsaklis and Sadrzadeh 2014). Our results (Section 4) show that event knowledge plays an important role in the compositional process and that it retains more or different information than what is generally encoded in distributional vectors.

2. Introducing MEDEA

Like in Chersoni, Lenci, and Blache (2017), the model is inspired by Memory, Unification and Control (MUC), proposed by Hagoort (2013, 2016) as a general model for the neurobiology of language. MUC incorporates three main functional components: i.) *Memory* corresponds to knowledge stored in long-term memory; ii.) *Unification* refers to the process of combining the units stored in *Memory* to create larger structures, with contributions from the context; and iii.) *Control* is responsible for relating language to joint action and social interaction. Similarly, our model distinguishes between:

- a **Distributional Event Graph** (DEG) that models a fragment of semantic memory activated by lexical units (Section 2.1);
- a **Meaning Composition Function** that dynamically integrates information activated from DEG to build a sentence semantic representation (Section 2.2)

2.1 Distributional Event Graph

In order to represent the GEK cued by lexical items during sentence comprehension, we explored a graph based implementation of a distributional model, for both theoretical and methodological reasons: in graphs, structural-syntactic information and lexical information can naturally coexist and be related, moreover vectorial distributional models often struggle with the modeling of dynamic phenomena, as it is often difficult to update the recorded information, while graphs are more suitable for situations where relations among items change overtime.

The data structure would ideally keep track of each event automatically retrieved from corpora, thus indirectly containing information about schematic or underspecified events, by abstracting over one or more participants from each recorded instance. Events are extracted from parsed sentences, using syntactic relations as an approxima-

tion of semantic roles (e.g., the subject relation for the agent, the direct object relation for the patient, etc.).¹

Given a lexical head (e.g., a verb or a noun), all its syntactic dependents are grouped together, similarly to the syntactic joint contexts for verb representation that were proposed by Chersoni et al. (2016).² More schematic events are also generated by abstracting from one or more event participants for every recorded instance (cf. Figure 1). We assume a very broad notion of *event*, as an *n*-ary relation between entities. Accordingly, an event can be a complex situation involving multiple participants, such as *The student reads a book in the library*, but also the association between an entity and a property expressed by the noun phrase *heavy book* (in accordance to what psychologists call *situation knowledge* or *thematic associations* (Binder 2016)). With respect to psycholinguistic research (McRae and Matsuki 2009), DEG can be regarded as a model of the generalized knowledge about events that can be derived from linguistic input, while in general GEK can be acquired from a richer array of inputs (e.g., including sensorimotor experience).

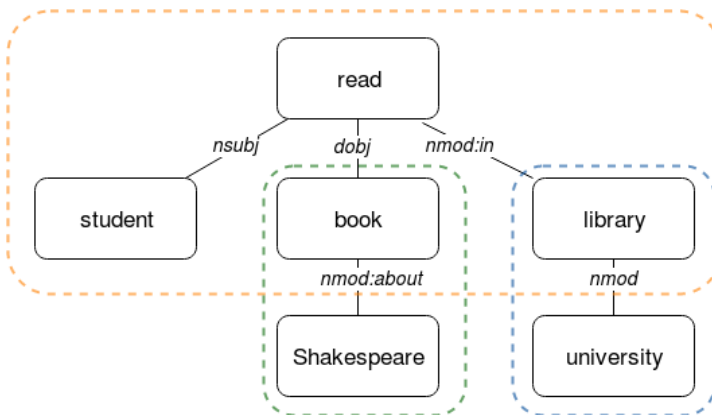


Figure 1

Reduced version of the parsing for the sentence *The student is reading the book about Shakespeare in the university library*. Three events are identified, each represented with a dotted box.

The nodes of DEG are lexical embeddings, and edges link lexical items participating to the same events (i.e., its syntagmatic neighbors, Figure 2). Edges are weighted with respect to the statistical salience of the event (i.e., the labeled link) given the item (i.e.,

- 1 We chose to use syntactic labels as a proxy for semantic relations for both general and practical reasons: from the practical point of view, syntactic annotation is much easier to obtain than semantic parsing, and many more resources are available with this kind of annotation. Moreover, dependency parsing, especially in the Universal Dependencies framework, provides a practical way to isolate relations between semantically full words such as nouns, verbs and adjectives, that are used here to cue the relevant subsets of DEG. In this sense, the choice poses some problems when it comes to more fine-grained semantic distinctions not easily captured through syntax (e.g., some prepositional complements may be ambiguous between *time* vs. *location* interpretation), but at the same time it offers a valuable opportunity and a simple way to deal with the notion of semantic role in a distributional approach.
- 2 *Syntactic joint contexts* are defined as an abstraction over joint contexts [Melamud et al. 2014], where each feature of the vector corresponds to a full argument constellation of a verb, to approximate the knowledge about typical event participants. For instance, a joint context for the verb *eat* in the *The dog eats the bone* is formed by both the subject *dog* and the direct object *bone*. The present work applies the same sort of notion to different categories than verbs and, more importantly, deals with the contextualization of the obtained representations.

Table 1

The five nearest paradigmatic and syntagmatic neighbors for the lexical item *book*, extracted from DEG.

Paradigmatic Neighbors	essay, story, novel, author, biography
Syntagmatic Neighbors	publish, write, read, child, series

the node). Weights, expressed in terms of a statistical association measure such as *Local Mutual Information*, determine the strength with which linguistic cues activate event information from the DEG. The resulting structure can therefore be seen as a weighted hypergraph, as it contains relations holding among groups of nodes, and a labeled multigraph, since each edge or hyperedge is labeled in order to represent the *syntactic pattern* holding in the group. Given the same group of words, in fact, different syntactic patterns are possible: for instance, considering the triplet *cat - chase - dog*, the pattern *nsubj - root - dobj* would indicate the event *The cat chases the dog*, while the pattern *dobj - root - nsubj* refers to the opposite situation where *The dog chases the cat*, and the pattern *nsubj - root - obl* could refer to a passive formulation such as *The cat is chased by the dog*.³ The weights are derived from co-occurrence statistics and measure the association strengths between event nodes. They are intended as salience scores that identify the most prototypical events associated with an entity (e.g., the typical actions performed by a student).

As graph nodes are embeddings, given a lexical cue w , DEG can be used to retrieve two kinds of information:

- the most similar nodes to w (i.e., its paradigmatic neighbors), using a vector similarity measure like the cosine⁴ (Table 1, top row);
- the closest associates of w (i.e., its syntagmatic neighbors), using the weights on the graph edges (Table 1, bottom row).

2.2 Meaning Composition Function

In MEDEA, we model sentence comprehension as the creation of a semantic representation SR (Figure 3), which includes two different yet interacting information tiers that are equally relevant in the overall representation of sentence meaning:

- *linguistic conditions* (LC) - a context-independent tier of meaning that accumulates the embeddings associated with the lexical items, as traditional compositional distributional models do;

³ The syntactic labels (e.g. *root*, *nsubj*, etc.) conform to the Universal Dependencies tagset available at <https://universaldependencies.org/>.

⁴ Cosine similarity is one of the most widely employed measures in vector space and quantifies the similarity of two non-zero vectors in terms of the angle between them:

$$\cos(\theta) = \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}} \quad (1)$$

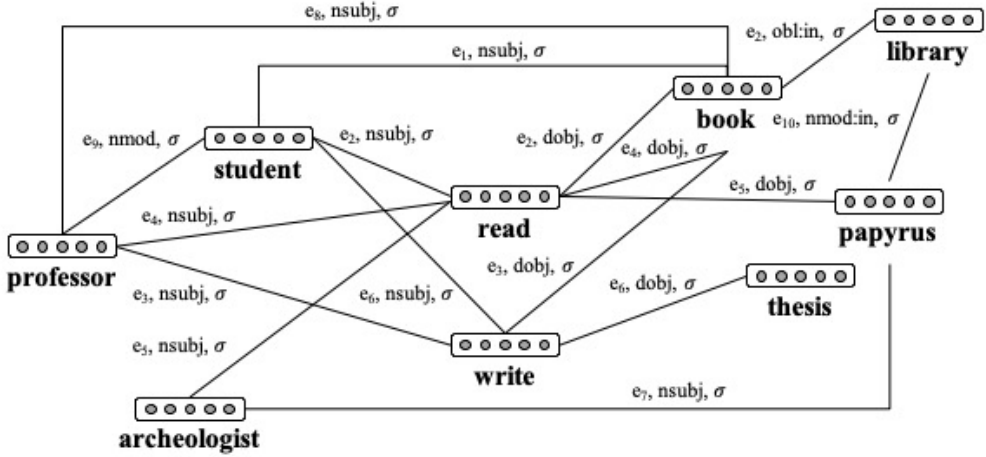


Figure 2

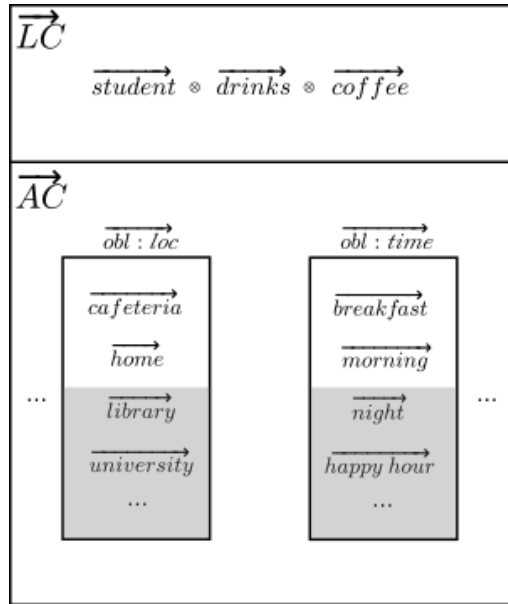
Toy example of DEG showing several instances of events, each represented by a sequence of co-indexed e . For example, e_2 corresponds to the event of students reading books in libraries, while e_1 and e_3 represent schematic events of students and professors performing some generic action on books (e.g., reading, consulting, studying, etc.). Each direct labeled edge is associated with its salience weight σ .

- *active context* (AC) - which aims at representing the most probable event, in terms of its participants, that can be reconstructed from DEG subsets cued by lexical items. More specifically, we assume that AC contains the embeddings activated from DEG by the single lexemes (or by other contextual elements) and integrated into a semantically coherent structure. The Active Context makes it possible to enrich the semantic content of the sentence with contextual information, predict other elements of the event, and generate expectations about incoming input. For instance, given the AC in Figure 3, we can predict that the student is most likely to be drinking a coffee at the cafeteria and that he/she is drinking it for breakfast or in the morning. The ranking of each element in AC depends on two factors: i.) its degree of activation by the lexical items, ii.) its overall coherence with respect to the information already available in the AC.

Let SR_{i-1} be the semantic representation built for the linguistic input w_1, \dots, w_{i-1} . When we process a new pair $\langle w_i, r_i \rangle$ with a lexeme w_i and syntactic role r_i :

1. LC in SR_{i-1} is updated with the embedding \vec{w}_i ;
2. AC in SR_{i-1} is updated with the embeddings of the syntagmatic neighbors of w_i extracted from DEG.

Figures 4 and 5 exemplify the update of the SR for the subject *student* with the information activated by the verb *drink*. The update process is defined as follows:

**Figure 3**

Sample SR for the sentence *The student drinks the coffee*. The LC includes the embeddings of the lexical items in the sentence and a generic composition function, while AC consists of lists of embeddings attached to a syntactic label: these have been activated from DEG and ranked by their salience with respect to the current content in the SR. Syntactic labels are taken as a surface approximation of their semantic role (e.g., the items listed under “obl:loc” are a set of possible locations of the event expressed by the sentence).

1. LC is represented with the vector \vec{LC} obtained from the combination of the embeddings of the words contained in the sentence. Therefore, when $\langle w_i, r_i \rangle$ is processed, the embedding \vec{w}_i is simply added to \vec{LC} ;
2. for each syntactic role r_i , AC contains a set of ranked lists (one for each processed pair that triggers that syntactic role) of embeddings corresponding to the most likely words expected to fill that role. For instance, the AC for the chunk *The student* in Figure 4 contains a list of the embeddings of the most expected main verbs and direct objects associated with *student*, a list of the embeddings of the most expected locations, etc. Each list of expected role fillers is itself represented with a centroid vector⁵ (e.g., $\vec{dob_j}$) of their k most prominent items (with k a model hyperparameter). For instance, setting $k = 2$, the $\vec{dob_j}$ centroid in the AC in figure 4 is built just from *book* and *research*; less salient elements (the gray areas in Figures 3, 4 and 5) are kept in the list of likely direct objects, but at this stage do not contribute to the centroid representing the expected fillers

⁵ A centroid vector is generally obtained as a (weighted) average of a set of vectors, namely

$\vec{X} = \frac{\sum_{i=0}^{|V|} p_i \vec{v}_i}{\sum_{i=0}^{|V|} p_i}$, where V is the set of vectors \vec{v}_i and p_i is a scalar representing the weight attributed to each vector.

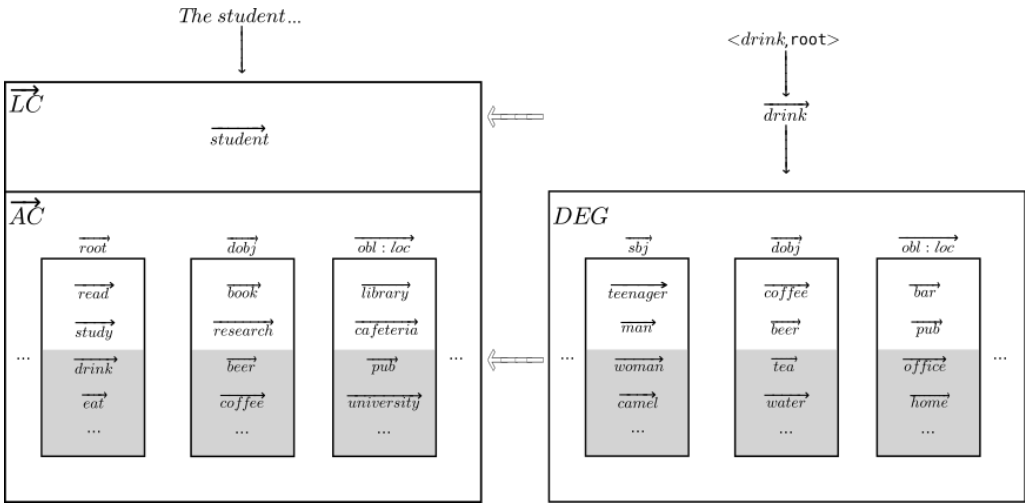
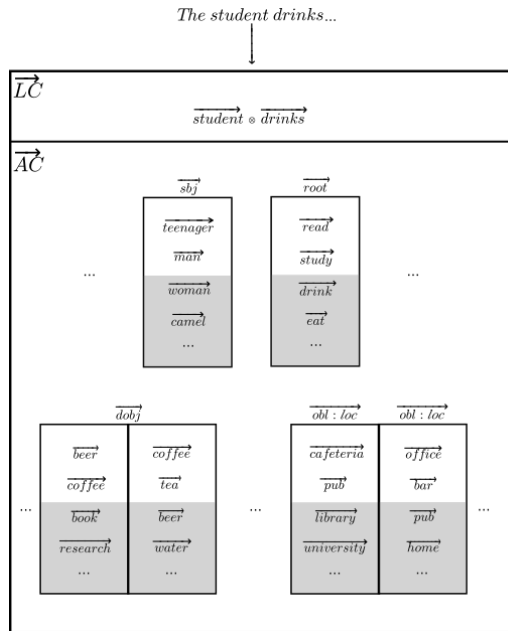


Figure 4
On the left, the SR generated after having processed the first chunk (i.e., *The student...*). On the right, the embedding and DEG subsets activated by the verb *drinks*.

for that role. AC is then updated with the DEG subset activated by the new lexeme w_i (e.g., the verb *drink*):

- the event knowledge activated by w_i for a given role r_i is ranked according to cosine similarity with the vector \vec{r}_i available in AC: in our example, the direct objects activated by the verb *drink* (e.g., \vec{beer} , \vec{coffee} , etc.) are ranked according to their cosine similarity to the \vec{dobj} vector of the AC;
- the ranking process works also in the opposite direction: the newly retrieved information is used to update the centroids in AC. For example, the direct objects activated by the verb *drink* are aggregated into centroids and the corresponding weighted lists in AC are re-ranked according to the cosine similarity with the new centroids, in order to maximize the semantic coherence of the representation. At this point, \vec{book} and $\vec{research}$, which are not as salient as \vec{coffee} and \vec{beer} in the *drinking* context, are downgraded in the ranked list and are therefore less likely to become part of the \vec{dobj} centroid at the next step.

The newly retrieved information is now added to the AC: as shown in Figure 5, once the pair $\langle \text{drink}, \text{root} \rangle$ has been fully processed, the AC contains lists for each triggered syntactic role, containing for example just one list for the *subj* role, which was only triggered by the verb, and two ranked lists for the *dobj* role, that was triggered by both previous elements. The whole AC is represented with the centroid vector \vec{AC} built out of a subset of the role vectors $\vec{r}_1, \dots, \vec{r}_n$ available in AC.

**Figure 5**

The original semantic representation SR for *The student...* is updated with the information activated by the verb, producing the SR for *The student drinks ...*. The new event knowledge is re-ranked with respect to the previous content of AC.

3. Experiments

3.1 Datasets and Tasks

Our aim is to evaluate the contribution of activated event knowledge in a sentence comprehension task. For this reason, among the several existing datasets concerning entailment or paraphrase detection, we chose RELPRON (Rimell et al. 2016), a dataset of subject and object relative clauses, and the transitive sentence similarity (TSS) dataset presented in Kartsaklis and Sadrzadeh (2014). These two datasets show an intermediate level of grammatical complexity, as they involve complete sentences (while other datasets include smaller phrases), but have fixed length structures featuring similar syntactic constructions (i.e., transitive sentences). The two datasets differ with respect to size and construction method.

RELPRON consists of 1,087 pairs, split in development (518 items) and test set (579 items), made up by a *target* noun labeled with a syntactic role (either *subject* or *direct object*) and a *property* expressed as *head noun* followed by a relative clause composed by a *verb* and a *nominal argument*. For instance, here are some example properties for the target noun *treaty*:

- (3) a. OBJ treaty: document that delegation negotiate
- b. SBJ treaty: document that grant independence

For each target t , the representations for the 518 properties in the dataset⁶ are built and ranked according to their similarity to t . Like Rimell et al. (2016), we use Mean Average Precision (henceforth MAP) to evaluate our models on RELPRON. Formally, MAP is defined as

$$MAP = \frac{1}{N} \sum_{i=1}^N AP(t_i) \quad (2)$$

where N is the number of target nouns in RELPRON, and $AP(t)$ is the Average Precision for target t , defined as:

$$AP(t) = \frac{1}{P_t} \sum_{k=1}^M Prec(k) \times rel(k) \quad (3)$$

Here, P_t is the number of correct properties for target t in the dataset, M is the total number of properties in the dataset, $Prec(k)$ is the precision at rank k , and $rel(k)$ is a function equal to one if the property at rank k is a correct property for t , and zero otherwise. Intuitively, $AP(t)$ will be 1 if, for the target t , all the correct properties associated to it are ranked in the top positions, and the value becomes lower when the correct properties are ranked farther from the head of the list.

We represented each property in RELPRON as a triplet $((hn, r), (w_1, r_1), (w_2, r_2))$ where hn is the head noun, w_1 and w_2 are the lexemes that compose the proper relative clause, and each element of the triplet is associated with its syntactic role in the property sentence.⁷

The TSS dataset consists of 108 pairs of transitive sentences, each annotated with human similarity judgments collected through the Amazon Mechanical Turk platform. Each transitive sentence is composed by a triplet *subject verb object*. Here are two pairs with high (4) and low (5) similarity scores respectively:

- (4) a. government use power
- b. authority exercise influence
- (5) a. team win match
- b. design reduce amount

Similarly to RELPRON properties, each sentence of the TSS is represented a triplet $((w_1, sbj), (w_2, root), (w_3, dobj))$. We built a compositional vector representation for both sentences of each item of the dataset, and then we measured the similarity between the resulting representations. Models are evaluated in terms of the Spearman correlation between the similarity scores and the human ratings.

⁶ Similarly to the Rimell et al. (2016) original paper, we only considered the items contained in the development set.

⁷ The relation for the head noun is assumed to be the same as the target relation (either *subject of direct object* of the relative clause).

3.2 MEDEA settings: data

We used the same corpora both to train the embeddings and to extract the syntactic relations for DEG. The training data comes from the concatenation of three dependency-parsed corpora: BNC (Leech 1992), ukWaC (Baroni et al. 2009) and a 2018 dump of the English Wikipedia, for a combined size of approximately 4 billion tokens. The corpora were parsed with Stanford CoreNLP (Manning et al. 2014).⁸

The embeddings associated to DEG lexical nodes were trained using the same parameters as in Rimell et al. (2016): we created lemmatized 100-dim vectors with *skip-gram* with *negative sampling* (SGNS) (Mikolov et al. 2013), setting minimum item frequency at 100 and context window size at 10.

3.3 MEDEA settings: DEG

We tailored the construction of DEG to the kind of simple syntactic structures required by the datasets (i.e., at most triplets of nodes), restricting it to the case of relations among pairs of event participants.

We included in the graph only events with a minimum frequency of 5 in the training corpora. The edges of the graph were weighted with *Smoothed LMI*. Given a triple composed by the words w_1 and w_2 , and a syntactic relation s linking them, we computed its weight by using a smoothed version of the Local Mutual Information (Evert 2004):

$$LMI_{\alpha}(w_1, w_2, s) = f(w_1, w_2, s) * \log\left(\frac{P(w_1, w_2, s)}{P(w_1) * P_{\alpha}(w_2) * P(s)}\right) \quad (4)$$

where the smoothed probabilities are defined as follows:

$$P_{\alpha}(x) = \frac{f(x)^{\alpha}}{\sum_x f(x)^{\alpha}} \quad (5)$$

Local Mutual information is often employed to balance the effects of frequency and to quantify the discrepancy between the chance of co-occurrence of two elements based on their individual and joint distributions. This type of smoothing, with $\alpha = 0.75$, was chosen to mitigate the bias of MI statistical association measures towards rare events (Levy, Goldberg, and Dagan 2015). While this formula only involves pairs (as only pairs were employed in the experiments), it is easily extensible to more complex tuples of elements.

3.3.1 LC

We implemented the additive model as a baseline, by considering only the LC tier of the SR and using addition as a composition function:

⁸ Note that in Rimell et al. (2016) the training corpus was a 2015 dump of Wikipedia.

$$\overrightarrow{LC} = \sum_{w \in sent} \vec{w} \quad (6)$$

3.3.2 AC content and re-ranking settings

In the present experiments, we did not use the predictions on non-expressed arguments to compute \overrightarrow{AC} . Moreover, we built the AC differently in the two tasks:

- as far as RELPRON is concerned, we restricted the evaluation to the representation of the target argument: for example, for the property *document that delegation negotiate*, the $\overrightarrow{AC}(sent)$ only contains the \overrightarrow{dobj} centroid;
- for the transitive sentences similarity dataset, the $\overrightarrow{AC}(sent)$ results from the summation of the centroids corresponding to the overtly filled roles in the sentence (i.e., *sbj*, *root*, *dobj*).

For each word in the dataset items, the top 50 associated words were retrieved from DEG. Both for the re-ranking phase and for the construction of the final representation, the event knowledge vectors (i.e., the role vectors \vec{r} and the \overrightarrow{AC} vector) are built from the top 20 elements of each weighted list. As detailed in Section 2.2, the ranking process in MEDEA can be performed forward and backward at the same time (i.e., the AC can be used to re-rank newly retrieved information and vice versa, respectively), but for simplicity we only implemented the forward ranking.

3.4 Scoring

We evaluated the performances of the LC component (i.e., our baseline), of the AC component alone and of the whole SR, as a summation of the first two scores.

Thus, in the case of RELPRON, given a *target* word in a sentence *sent*, the score for MEDEA is computed as a summation of two cosine scores:

$$score(target, sent) = cosine(\overrightarrow{target}, \overrightarrow{LC}(sent)) + cosine(\overrightarrow{target}, \overrightarrow{AC}(sent)) \quad (7)$$

whereas in the case of the transitive sentences similarity dataset, given two sentences s_1, s_2 the score for MEDEA is computed as:

$$score(s_1, s_2) = cosine(\overrightarrow{LC}(s_1), \overrightarrow{LC}(s_2)) + cosine(\overrightarrow{AC}(s_1), \overrightarrow{AC}(s_2)) \quad (8)$$

In all settings, we assume the model to be aware of the syntactic parse of the test items. In the transitive sentences similarity dataset, word order fully determines the syntactic constituents, as the sentences are always in the *subject verb object* order. In RELPRON, on the other hand, the item contains information about the relation that is being tested: in the *subject* relative clauses, the properties always show the *verb* followed by the *argument* (e.g., *telescope: device that detects planets*), while in the *object* relative clauses the properties always present the opposite situation (e.g., *telescope: device that observatory has*).

4. Results and Discussion

4.1 RELPRON

Given the targets and the composed vectors of all the definitions in RELPRON, we assessed the cosine similarity of each pair and computed the Mean Average Precision scores shown in Table 2.

Table 2

The table shows results in terms of MAP for the development subset of RELPRON.

	RELPRON		
	LC	AC	LC+AC
verb	0,18	0,18	0,20
arg	0,34	0,34	0,36
hn+verb	0,27	0,28	0,29
hn+arg	0,47	0,45	0,49
verb+arg	0,42	0,28	0,39
hn+verb+arg	0,51	0,47	0,55

Following the original evaluation in Rimell et al. (2016), we tested six different combinations for each composition model: the verb only, the argument only, the head noun and the verb, the head noun and the argument, the verb and the argument and all three of them. In all cases but one, the models built on the complete SR (i.e., involving the LC level and the AC level) show significant improvements, outperforming the simple additive baseline. Most interestingly, the models involving only the AC tier of the semantic representation still show comparable performances to the baseline.

The only model that lags behind is the *verb+arg* model. As also shown in Rimell et al. (2016), models involving only the sum of lexical vectors show balanced results (Table 3). Things are instead different for the AC component. Here, the composition of event knowledge elicited by verb and argument seems much better at predicting the object than the subject.

Table 3

The table shows MAP results, for each model involving only the LC component, for subject and object relations separately.

LC	verb	arg	hn+verb	hn+arg	verb+arg	hn+verb+arg
<i>subject</i>	0,21	0,44	0,30	0,55	0,49	0,60
<i>object</i>	0,20	0,39	0,30	0,52	0,48	0,59
Δ	0,01	0,06	0,00	0,04	0,01	0,01

Table 4

The table shows MAP results, for each model involving only the AC component, for subject and object relations separately.

AC	verb	arg	hn+verb	hn+arg	verb+arg	hn+verb+arg
<i>subject</i>	0,19	0,41	0,29	0,47	0,22	0,48
<i>object</i>	0,19	0,34	0,29	0,51	0,38	0,52
Δ	0,00	0,06	0,00	-0,04	-0,16	-0,04

One relevant parameter of the models is that they work in the linear order in which words are found in the sentence. The *verb+arg* model, therefore, works differently when run on *subject* clauses than on *object* clauses. In the *subject* case, in fact, the verb is found first, and then its expectations are used to reweigh the ones of the object. In the *object* case, on the other hand, things go the opposite way: at first the subject is found, and then its expectations are used to reweigh the ones of the verb (see table 5). When testing the same model, but in reverse order of activation (the second word of the property and then the first one), we find opposite results, with a MAP of 0.41 for *subjects* and 0.21 for *objects*. It seems that, when arguments, which are nouns, are encountered first, event knowledge is more precise and better at predicting the target. This is in line with the fact that *arguments* alone perform better than *roots* alone, and could be related to the fact that verb perform in general distributionally worse than nouns on standard similarity tasks.

Table 5
The table shows the differences between standard linear order (first row) and reverse order (second row) for *subject* and *object* relative clauses. Values in bold refer to the models that show best performances.

	<i>subject</i> clause	<i>object</i> clause
w_1 w_2 order	V - O	S - V
w_2 w_1 order	O - V	V - S

4.2 Transitive sentences dataset

For the transitive sentences dataset, we evaluated the correlation of our scores with human ratings with Spearman’s ρ . The similarity between a pair of sentences s_1, s_2 is defined as the cosine between their LC vectors plus the cosine between their AC vectors. We tested seven different combinations for each composition model, evaluating the contribution of each subset of the sentence (i.e., *subject* alone, *verb* alone, *subject+verb*, etc., up to the full sentence).

MEDEA is in the last column of Table 6 and again outperforms simple addition in most cases. Event knowledge alone (i.e., AC column of Table 6) outperforms the baseline in the *sbj* and *root* models, suggesting that information on event knowledge is not properly encoded in distributional vectors, and possibly captures different aspects of compositional meaning. Except for the case of *sbj+root*, the models involving event knowledge in AC always improve the baselines.

5. Conclusion

We provided a basic implementation of a meaning composition model, which aims at being incremental and cognitively plausible. While still relying on vector addition, our results suggest that distributional vectors do not encode sufficient information about event knowledge, and that, in line with psycholinguistic results, activated GEK plays an important role in building semantic representations during online sentence processing.

Our ongoing work focuses on refining the way in which this event knowledge takes part in the processing phase and testing its performance on more complex datasets: while both RELPRON and the transitive sentences dataset provided a straightforward mapping between syntactic label and semantic roles, more naturalistic datasets show a

Table 6

The table shows results in terms of Spearman's ρ on the transitive sentences dataset. p -values are not shown because they are all equally significant ($p < 0.01$).

	transitive sentences dataset		
	LC	AC	LC+AC
sbj	0.432	0.475	0.482
root	0.525	0.547	0.555
obj	0.628	0.537	0.637
sbj+root	0.656	0.622	0.648
sbj+obj	0.653	0.605	0.656
root+obj	0.732	0.696	0.750
sbj+root+obj	0.732	0.686	0.750

much wider range of syntactic phenomena that would allow us to test how expectations jointly work on the event structure, both at the syntactic level and with respect to the semantic roles filled by participants. Similarly, we will consider more complex tasks such as entailment or inference, for which a variety of datasets are available in literature, in order to evaluate the model's performances on broader language understanding benchmarks.

References

- Asher, Nicholas, Tim Van de Cruys, Antoine Bride, and Márta Abrusán. 2016. Integrating Type Theory and Distributional Semantics: A Case Study on Adjective–Noun Compositions. *Computational Linguistics*, 42(4):703–725.
- Baroni, Marco, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in Space: A Program of Compositional Distributional Semantics. *Linguistic Issues in Language Technology*, 9(6):5–110.
- Baroni, Marco, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.
- Binder, Jeffrey R. 2016. In Defense of Abstract Conceptual Representations. *Psychonomic Bulletin & Review*, 23:1096–1108.
- Blacoe, William and Mirella Lapata. 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*, pages 546–556, Jeju Island, Korea, July 12–14. Association for Computational Linguistics.
- Chersoni, Emmanuele, Alessandro Lenci, and Philippe Blache. 2017. Logical Metonymy in a Distributional Model of Sentence Comprehension. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 168–177, Vancouver, Canada, August 3–4.
- Chersoni, Emmanuele, Enrico Santus, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. 2016. Representing Verbs with Rich Contexts: An Evaluation on Verb Similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1967–1972, Austin, TX, USA, November 1–5.
- Coecke, Bob, Stephen Clark, and Mehrnoosh Sadrzadeh. 2010. Mathematical foundations for a compositional distributional model of meaning. Technical report.
- Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 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*, pages 670–680, Copenhagen, Denmark, September, 7–11.
- Elman, Jeffrey L. 2011. Lexical knowledge without a lexicon? *The mental lexicon*, 6(1):1–33.
- Erk, Katrin and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Honolulu, Hawaii, USA, October 25–27. Association for Computational Linguistics.

- Evert, Stefan. 2004. *The Statistics of Word Cooccurrences Word Pairs and Collocations*. Ph.D. thesis, University of Stuttgart.
- Hagoort, Peter. 2013. MUC (Memory, Unification, Control) and Beyond. *Frontiers in Psychology*, 4(JUL):1–13.
- Hagoort, Peter. 2016. MUC (Memory, Unification, Control): A Model on the Neurobiology of Language beyond Single Word Processing. In Gregory Hickok and Steve Small, editors, *Neurobiology of Language*, volume 28. Elsevier, Amsterdam, pages 339–347.
- Hagoort, Peter and Jos van Berkum. 2007. Beyond the sentence given. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):801–811.
- Hare, Mary, Michael Jones, Caroline Thomson, Sarah Kelly, and Ken McRae. 2009. Activating Event Knowledge. *Cognition*, 111(2):151–167.
- Jackendoff, Ray. 1997. *The Architecture of the Language Faculty*. MIT Press, Cambridge, MA.
- Kartsaklis, Dimitri and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*. Kyoto, Japan, 4–6th June.
- Kiros, Ryan, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 3294–3302.
- Kuperberg, Gina R. and T. Florian Jaeger. 2016. What do we mean by prediction in language comprehension? *Language, cognition and neuroscience*, 31(1):32–59.
- Leech, Geoffrey Neil. 1992. 100 Million Words of English: The British National Corpus (BNC).
- Lenci, Alessandro. 2018. Dynamic Distributional Semantics. Unpublished manuscript.
- Levy, Omer, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.
- Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL 2014, the 52nd Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, pages 55–60, Baltimore, MD, USA, 22–27 June.
- McRae, Ken and Kazunaga Matsuki. 2009. People Use their Knowledge of Common Events to Understand Language, and Do So as Quickly as Possible. *Language and Linguistics Compass*, 3(6):1417–1429.
- Melamud, Oren, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret. 2014. Probabilistic modeling of joint-context in distributional similarity. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 181–190, Baltimore, Maryland, USA, 26–27 June.
- Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119, Lake Tahoe, NV, USA, December 5–10.
- Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition.
- Partee, Barbara H. 1984. Compositionality. In *Varieties of Formal Semantics*. Foris, Dordrecht, pages 281–311.
- Rimell, Laura, Jean Maillard, Tamara Polajnar, and Stephen Clark. 2016. RELPRON: A Relative Clause Evaluation Data Set for Compositional Distributional Semantics. *Computational Linguistics*, 42(4):661–701.

Multi-source Transformer for Automatic Post-Editing of Machine Translation Output

Amirhossein Tebbifakhr*

Fondazione Bruno Kessler

Università di Trento

Matteo Negri**

Fondazione Bruno Kessler

Marco Turchi†

Fondazione Bruno Kessler

Automatic post-editing (APE) of machine translation (MT) is the task of automatically fixing errors in a machine-translated text by learning from human corrections. Recent APE approaches have shown that best results are obtained by neural multi-source models that correct the raw MT output by also considering information from the corresponding source sentence. In this paper, we pursue this objective by exploiting Transformer (Vaswani et al. 2017), the state-of-the-art architecture in MT. Our approach presents several advantages over previous APE solutions, both from the performance perspective and from an industrial deployment standpoint. Indeed, besides competitive results, our Transformer-based architecture is faster to train (thanks to parallelization) and easier to maintain (thanks to the reliance on a single model rather than a complex, multi-component architecture). These advantages make our approach particularly appealing for the industrial sector, where scalability and cost-efficiency are important factors, complementary to pure performance. Besides introducing a novel architecture, we also validate the common assumption that training neural APE systems with more data always results in stronger models. Along this direction, we show that this assumption does not always hold, and that fine-tuning the system only on small in-domain data can yield higher performance. Furthermore, we try different strategies to better exploit the in-domain data. In particular, we adapt reinforcement learning (RL) techniques to optimize our models by considering task-specific metrics (i.e. BLEU and TER) in addition to maximum likelihood. Our experiments show that, alone, the multi-source approach achieves slight improvements over a competitive APE system based on a recurrent neural network architecture. Further gains are obtained by the full-fledged system, fine-tuned on in-domain data and enhanced with RL optimization techniques. Our best results (with a single multi-source model) significantly improve the performance of the best (and much more complex) system submitted to the WMT 2017 APE shared task.

1. Introduction

Recent advances in Machine Translation (MT) have made the post-editing of raw MT output more cost-efficient in industry settings compared to human translation from scratch. In this *translation as post-editing* workflow, firstly the source text is translated

* Fondazione Bruno Kessler, Via Somarive 18, 38123 Povo (Trento), Italy. E-mail: atebdifakhr@fbk.eu

** Fondazione Bruno Kessler, Via Somarive 18, 38123 Povo (Trento), Italy. E-mail: negri@fbk.eu

† Fondazione Bruno Kessler, Via Somarive 18, 38123 Povo (Trento), Italy. E-mail: turchi@fbk.eu

by means of an MT system. Then, the machine-translated text is revised by a human expert in order to correct possible errors. Although the reliance on high-quality MT systems reduces the amount of effort needed by professional translators, state-of-the-art MT output is not yet perfect: systematic errors may still occur, which require repeated human corrections of similar mistakes. Automatic Post-Editing (APE) aims to automatize this process and, in doing so, to speed-up, reduce the workload and the overall costs of professional translation.

The APE task can be cast as a “monolingual translation” problem (i.e. the translation from raw MT output into publishable material), in which a model is trained, in a supervised fashion, on datasets comprising (*source-text*, *MT-output*, *human_post-edit*) triplets. Cast in this way, the problem can be approached thanks to the wealth of data daily produced by modern industry workflows based on the translation as post-editing paradigm. In terms of approaches, APE research followed a similar path to that of MT. Early approaches based on the statistical paradigm (Simard et al. 2007) were recently replaced by more advanced and effective neural solutions. Among them, the most effective ones (Chatterjee et al. 2015; Pal et al. 2016) operate by looking not only at the MT output to be corrected but also at the corresponding source text (i.e. correction actions are conditioned to both the texts) in order to leverage more information, resolve potential ambiguities, and eventually return more reliable corrections. However, a drawback of these state-of-the-art APE solutions is the reliance on pipelined architectures (Bojar et al. 2017), whose complexity raises training/maintenance issues and eventually reduces their usability. Indeed, current top systems typically rely on ensembling multiple recurrent neural networks (RNNs) and performing a final re-ranking step (Chatterjee et al. 2017) to select the most promising correction hypothesis. Though competitive, such architectures require training and maintaining multiple components, involving costs that reduce their appeal from the industry perspective.

To overcome these issues, in this paper we first propose a single multi-source APE system based on Transformer (Vaswani et al. 2017), the state-of-the-art architecture in MT. Our experiments show that this system can reach state-of-the-art performance on the benchmark released for the WMT 2017 APE shared task (Bojar et al. 2017). Beside the better performance, our system has two advantages compared to previous approaches. First, it is faster to train compared to previous RNN-based solutions. This is due to the fact that, in contrast to the auto-regressive nature of RNNs, in the Transformer architecture all the positions in the sentence are processed in parallel. Second, it is easier to maintain, since only one single model is trained in an end-to-end fashion, and there is no need for optimizing different components interacting with each other.

Moving a step forward in our research, we then focus on the data dimension. Indeed, although neural approaches achieve state-of-the-art results (in APE like in MT), they need to be trained on huge amounts of data. Prior work heavily built on a “the more the better” assumption (Bojar et al. 2017). Following this assumption, all the available training data are used and possibly augmented with large synthetic datasets (Junczys-Dowmunt and Grundkiewicz 2016). The way data are exploited, however, is still an unexplored problem, leaving large room for research on how to optimize their use towards better performance. Along this direction, we investigate different ways to combine gold, in-domain data with large, sub-optimal synthetic corpora. In particular, focusing on model optimization, we adapt to APE different ways to maximize the exploitation of small in-domain corpora rather than simply relying on a brute force processing of all the available data. To this aim, we choose optimization strategies that are drawn from Reinforcement learning (Ranzato et al. 2016; Shen et al. 2016). The advantage of these strategies instead of maximum likelihood is twofold. First, to make

the optimization process more reliable, they directly target task-specific, reference-based evaluation metrics (BLEU and TER) instead of maximizing the likelihood of the training data, which can have low correlation with these metrics. Second, being more directly driven by the final evaluation metrics, they are more suitable to preserve the style of the reference translations. This property is particularly important in APE, where the system should mimic the behaviour of human post-editors (i.e. perform the minimum amount of edit operations required to fix the MT errors), without “over-correcting” the machine-translated text with unnecessary complete rephrasing. Discouraging the system to drastically change the MT output (even with post-edits that are *per se* correct) is crucial to avoid penalizing the system by automatic evaluation metrics based on references built from minimal human corrections.

Our main contributions can be summarized as follows:

- We introduce a neural, multi-source APE system based on the Transformer architecture;
- We conduct a set of experiments in order to analyze the effect of using synthetic and in-domain training data to build a neural APE model;
- We explore different reinforcement learning approaches in order to maximize the usefulness of in-domain data during training;
- We evaluate our multi-source Transformer-based system and the proposed enhancements on a shared evaluation benchmark, on which we achieve state-of-the-art results.

This research extends previous work by the same authors (Tebbifakhr et al. 2018), by adding the analysis of using different type of data and different optimization strategies, which ends to state-of-the-art results in the APE task. The paper is organized as follows. In Section 2, we review the previous works on APE. In Section 3, we describe our multi-source architecture based on Transformer and introduce different model optimization strategies drawn from reinforcement learning. In Section 4, we describe our experimental setup and model configuration. In Section 5, we discuss the results of our experiments. Finally, Section 6 summarizes the paper with conclusions.

2. Related Work

The APE task was introduced for the first time by Knight and Chander (1994) to automatically insert the correct articles in Japanese to English translation. Other more recent works adopted rule-based approaches (Rosa, Mareček, and Dušek 2012), but they gained limited attention, due to involving extensive manual work in defining the rules.

A statistical approach has been proposed for the first time in (Simard et al. 2007), in which the problem is cast as a “monolingual translation” task. In particular, raw MT output is “translated” into better translations by means of a phrase-based MT (PB-SMT) system trained on (MT_output, post-edited_output) pairs. They showed that APE components trained on human corrections of machine translated texts yield substantial improvements to the original MT system output. This results in further investigation of using PBSMT systems as an APE module for different language directions, over different underlying MT system architectures, in different domains (Isabelle, Goutte, and Simard 2007; Dugast, Senellart, and Koehn 2009; Lagarda et al. 2009; Béchara et al. 2012; Rosa, Mareček, and Tamchyna 2013). In (Isabelle, Goutte, and Simard 2007) the

authors used an APE module for domain adaptation: the idea is to learn from human corrections not only how to fix MT errors, but also how to adequately translate in a specific target domain. It has shown to be useful for reducing time, effort and the overall costs of human translation in industry environments (Aziz, Castilho, and Specia 2012). Recently, the advent of deep learning (in particular neural machine translation – NMT) caused a shift from the statistical approaches to more powerful neural approaches. (Pal et al. 2016; Junczys-Dowmunt and Grundkiewicz 2016) are among the first works in APE relying on Neural Machine Translation (NMT) models. These papers rely on an encoder-decoder architecture with attention model. On different datasets, neural APE models were able to significantly improve the outputs of PBSMT systems. Building on these positive results, this paper takes a step forward by exploiting Transformer (Vaswani et al. 2017) (i.e. the state of the art architecture for MT) to approach the APE task.

Although casting the APE task as a monolingual translation gained lots of attention, it has a major drawback, which is disregarding the information about the source text. Indeed, part of the information in the source text can be lost by errors made by the MT system, which cannot be rectified by only attending on the MT output. To cope with this issue, Béchara, Ma, and van Genabith (2011) proposed a “source context-aware” variant of the work by Simard et al. (2007). This approach creates a new input sentence representation by combining each MT word with its corresponding source word. When tested with a PBSMT APE system, it results in better performance than using only the MT sentence. In neural APE, Junczys-Dowmunt and Grundkiewicz (2016) use a log-linear combination of two NMT models (source-text - post-edited_output and MT_output - post-edited_output) for translating source and raw MT output to human post-edited text, both using (Bahdanau, Cho, and Bengio 2015) architecture. In (Chatterjee et al. 2017), they show that the best results can be obtained by a single APE system that uses two different encoders to exploit the source and MT sentences in APE. In this paper, we further explore the potential of the multi-source approach by leveraging the Transformer, a more powerful NMT architecture.

More recently, addressing the problem of over-correction in APE, (Chatterjee et al. 2018) examine different strategies to combine Quality Estimation (QE) and APE. The over-correction problem happens when APE system tends to rephrase an already good translation. To prevent this problem, they propose three different approaches for integrating QE and APE: *i*) QE as an activator, which decides if the MT sentence needs to be post-edited or not, *ii*) QE as a selector, which selects the best output between the MT output and the post-edited text, and *iii*) QE as a guidance, which identifies problematic parts of the translation for post-editing.

The parameters of most NMT systems and consequently neural APE models are optimized by maximizing the likelihood of the training data. Indeed, a token-level cross-entropy loss function is defined to maximize the probability of each token in the target sequence. However, the performance of these systems is evaluated using sequence-level evaluation metrics such as BLEU (Papineni et al. 2002) and TER (Snover et al. 2006). To cope this discrepancy, different reinforcement learning methods such as REINFORCE (Ranzato et al. 2016), actor-critic (Bahdanau et al. 2016) and minimum risk training (Shen et al. 2016) have been used to directly maximize these sequence-level metrics for sequence generation tasks. All the approaches show that integrating the evaluation metric in the optimization process results in improvement in performance. These approaches that have been extensively used in NMT, have not been tested in APE. Exploring them, together with analysing their effectiveness in different data conditions

is another contribution of this paper. As a final contribution, we present state-of-the-art results obtained by our architecture on a shared evaluation benchmark.

3. A Neural APE System Based on Transformer

In this section we introduce: *i)* the Transformer architecture, *ii)* its multi-source extension and *iii)* optimization techniques, drawn from reinforcement learning, targeting the integration of task-specific metrics in the computation of the loss. Altogether, these elements represent the core of our approach to neural APE.

3.1 Transformer and its Multi-Source Extension

As discussed in the previous sections, current approaches to APE as a “monolingual translation” task rely on state-of-the-art architectures developed for neural MT. Typically, they employ deep recurrent networks (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015) in the so-called encoder-decoder framework. In this framework, a sequence of words $[x_1, x_2, \dots, x_n]$ is given to an encoder, which maps it to a sequence of continuous representations, i.e. the hidden state of the encoder. At each time step, based on these continuous representations and the generated word in the previous time step, a decoder generates the next word. This process continues until the decoder generates the end-of-sentence word. More formally, at each time step the decoder predicts the next word y_n , given the context vector c and the previously predicted words $y_{<n}$ by defining a probability over the translation y as follows:

$$p(\mathbf{y}) = \prod_{n=1}^N p(y_n | y_{<n}, c) \quad (1)$$

The context vector c is a weighted sum computed over the hidden states of the encoder. The weights used to compute the context vector are obtained by a network called attention model that finds an alignment between the target and source words (Bahdanau, Cho, and Bengio 2015). From an efficiency standpoint, a major drawback of these approaches is that, at each time step, the decoder needs the hidden state of the previous time step, thus hindering parallelization. Other approaches have been proposed to avoid this sequential dependency (e.g. using convolution as a main building block) and make parallelization possible (Gehring et al. 2017; Kalchbrenner et al. 2016). However, although they can avoid the recurrence, they are not able to properly learn the long term dependencies between words.

The Transformer architecture introduced in (Vaswani et al. 2017), set a new state-of-the-art in NMT by completely avoiding both recurrence and convolution. Since the model does not leverage word-order information, it adds positional encoding to the input word embeddings to enable capturing word order. This aspect is particularly important since, in contrast to the auto-regressive nature of RNNs, it allows Transformer to process all the input positions in parallel, with considerable savings in computation time.

In Transformer, the attention employed is a multi-headed self-attention, which is a mapping from $(query, key, value)$ tuples to an output vector. The self-attention is defined

as follows:

$$SA(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2)$$

where Q is the query matrix, K is the key matrix and V is the value matrix, d_k is the dimensionality of the queries and keys, and SA is the computed self-attention. The multi-head attention (MH) is computed as follows:

$$MH(Q, K, V) = \text{Concat}(head_1, \dots, head_h) W^O \quad (3)$$

where h is the number of attention layers (also called “heads”), $head_i$ is the self-attention computed over the i^{th} attention layer and W^O is the parameter matrix of dimension $hd_v * d_{model}$. The encoder layers consist of a multi-head self-attention, followed by a position-wise feed forward network. In the self-attention, the queries, keys and values matrices come from the previous layer. In the decoder, the layers have an extra encoder-decoder multi-head attention after the multi-head self-attention, where the key and value matrices come from the encoder and the query matrix comes from the previous layer in the decoder. Also, inputs to the multi-head self-attention in the decoder are masked in order to not attend to the next positions. Finally, a softmax normalization is applied to the output of the last layer in the decoder to generate a probability distribution over the target vocabulary.

In order to encode the source sentence in addition to the MT output, we employ the multi-source method (Zoph and Knight 2016), wherein the model is comprised of two separate encoders (with a different set of parameters) that respectively provide continuous representations the source sentence and the MT output. The output of the two encoders is concatenated and passed as the key in the attention. This helps for a better representation, in turn leading to more effective attention during decoding time.

3.2 Integrating Task-Specific Metrics in Loss Computation

While in the previous section we overviewed Transformer and its multi-source extension introduced in (Tebbifakhr et al. 2018), in this section we present a further extension of the approach. In particular, targeting a more effective model optimization, we discuss the application of different learning strategies based on reinforcement learning.

In general, neural MT systems are optimized to maximize the likelihood of training data (i.e. Maximum Likelihood Estimation – MLE). In order to maximize the likelihood of the given parallel corpus $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}_{s=1}^S$, they use token-level cross-entropy as the objective function. Therefore, the loss function is defined as follows:

$$\mathcal{L}_{MLE} = - \sum_{s=1}^S \sum_{n=1}^{N^{(s)}} \log p(\mathbf{y}_n^{(s)} | \mathbf{y}_{<n}^{(s)}, \mathbf{x}) \quad (4)$$

where $N^{(s)}$ is the length of s -th target sentence and $p(\mathbf{y}_n^{(s)} | \mathbf{y}_{<n}^{(s)}, \mathbf{x})$ is the probability of generating the n -th word of $\mathbf{y}^{(s)}$ given by preceding ground-truth words and $\mathbf{x}^{(s)}$.

However, as pointed out by (Ranzato et al. 2016), model optimization by maximizing the likelihood of training data has two main drawbacks. One is the so-called “exposure bias”: while models are trained only on the distribution of the training data, the

generation of target words at test time is conditioned to the previous model predictions. In this way, possible decoding errors at previous time steps will have significant impact on the following steps making the output generation diverge to completely wrong sentences (especially in the case of long input sentences). The other is that maximizing the probability of the next correct word based on the cross-entropy loss computed on previous word-level errors may have a low correlation with the metrics actually used for evaluation. To avoid these problems, better approaches should maximize the task-specific evaluation metrics that actually quantify translation quality. These, in the case of APE, are BLEU (Papineni et al. 2002) and TER (Snover et al. 2006). However, since these metrics are not parameterized w.r.t. to the model parameters, it is impossible to use common gradient descent for model optimization. This problem, which is well-known in MT, becomes particularly severe in APE, where automatic evaluation metrics against minimal post-edits (i.e. conservative “human-like” post edits) would penalise system’s tendency to “over-correct” acceptable portions of the input sentence. Indeed, in APE, the goal is not to fully re-translate the MT output, which might certainly result in a good translation. Rather, the goal is to mimic post-editors’ behaviour so to perform only the necessary corrections, leaving the correct parts of the MT output untouched. Maximizing BLUE (or minimizing TER) with respect to human post-edits will push the system towards a conservative correction strategy rather than the potentially more aggressive maximization of translations’ probability.

In order to directly maximize these evaluation metrics, different approaches have been introduced, which mainly use reinforcement learning techniques. In RL methods for MT (Ranzato et al. 2016; Shen et al. 2016), the parameters of the MT system define a *policy* that chooses an *action*, i.e. generating a translation candidate \hat{y} , and gets a *reward* $\Delta(\hat{y})$ according to that action, i.e. a measure of output goodness based on the evaluation metric. The loss function in RL for MT is defined as expected reward, which has to be maximized:

$$\begin{aligned}\mathcal{L}_{RL} &= \sum_{s=1}^S E_{\hat{y} \sim p(\cdot | \mathbf{x}^{(s)})} \Delta(\hat{y}) \\ &= \sum_{s=1}^S \sum_{\hat{y} \in \mathbf{Y}} p(\hat{y} | \mathbf{x}^{(s)}) \Delta(\hat{y})\end{aligned}\tag{5}$$

where \mathbf{Y} is the set of all possible translation candidates. However, since the size of this set is exponentially large, it is practically impossible to compute the expected reward. To bypass this problem, in REINFORCE (Ranzato et al. 2016) the expected reward is estimated by random sampling only one candidate from this set.

$$\hat{\mathcal{L}}_{RL} = \sum_{s=1}^S p(\hat{y} | \mathbf{x}^{(s)}) \Delta(\hat{y}), \hat{y} \sim p(\cdot | \mathbf{x}^{(s)})\tag{6}$$

Alternatively, in Minimum Risk Training (MRT – (Shen et al. 2016)), this expected reward is estimated by subsampling a candidates set.

$$\hat{\mathcal{L}}_{RL} = \sum_{s=1}^S \sum_{\hat{y} \in \mathcal{S}} q(\hat{y} | \mathbf{x}^{(s)}) \Delta(\hat{y})\tag{7}$$

Table 1

Statistics for the synthetic and in-domain datasets.

train			development	test	
synthetic 4M	synthetic 500K	in-domain	in-domain	in-domain 2016	in-domain 2017
4,391,180	526,368	23,000	1,000	2,000	2,000

where $q(\hat{\mathbf{y}}|\mathbf{x}^{(s)})$ is the normalized probability of the each hypothesis in the subsampled set \mathcal{S} :

$$q(\mathbf{y}|\mathbf{x}^{(s)}) = \frac{p(\mathbf{y}|\mathbf{x}^{(s)})}{\sum_{\mathbf{y}' \in \mathcal{S}} p(\mathbf{y}'|\mathbf{x}^{(s)})} \quad (8)$$

In the remainder of this paper, we will explore the application of both the methods to optimize our APE models on small in-domain data. Although these loss functions allow to directly optimize the model parameters to maximize task-specific evaluation metrics, recent studies on reinforcement learning for NMT have shown that their combination with MLE can yield further improvements (Wu et al. 2018). Moving a step forward, we will hence build on these findings and, in addition to RL-based loss functions, we will also experiment with their combination with MLE, so to obtain a combined loss function that considers both intrinsic word probabilities and extrinsic task-specific evaluation criteria:

$$\mathcal{L} = \mathcal{L}_{RL} + \mathcal{L}_{MLE} \quad (9)$$

4. Experiment Setup

In this section, we outline the data used for training and evaluating our APE system. Then, we describe the metrics used for evaluation. Finally, we give details about our baselines, the evaluated models and their specific setting.

4.1 Data

For the sake of a fair comparison with the best performing system at the WMT 2017 APE shared task (Chatterjee et al. 2017), we use the same training, development and test WMT datasets. The training data consists of three different corpora. One of them is released by the task organizers and contains 23K triplets from the Information Technology domain. The other two are synthetic data created by (Junczys-Dowmunt and Junczys-Dowmunt 2017). They respectively contain $\sim 4\text{M}$ and $\sim 500\text{K}$ English-German triplets generated by a round-trip translation process. In this process, a German-to-English and an English-to-German phrase-based translation models are respectively used to first translate German data into English, and then to translate the obtained output back into German. The original German monolingual data are considered as post-edits, the English translated data are considered as source sentences, and the German back-

translated data are considered as machine translation outputs. The development set is the one released for WMT 2017 APE shared task, which contains 1K in-domain triplets. We evaluate our models using the two test sets released for WMT 2016 and 2017 APE shared tasks, each containing 2K in-domain triplets. Table 1 summarizes the statistics of the datasets. To avoid unknown words and to keep under control the vocabulary size, we apply byte pair encoding (Sennrich, Haddow, and Birch 2016) to all the data using 32K rules.

4.2 Evaluation Metrics

For evaluation, we use the two official metrics of the WMT APE task, namely: *i)* TER (Snover et al. 2006) which is based on edit distance and *ii)* BLEU, which is the geometric mean of n-gram precision (Papineni et al. 2002). They are both applied on tokenized and true-cased data.

4.3 Terms of Comparison

We compare the performance of our Transformer models with two baselines.

Our first baseline (**MT Baseline**) is the official baseline used in the APE shared tasks; it mimics the behaviour of a “*do-nothing*” APE model that leaves all the original MT outputs untouched. In other words, leaving them unmodified, it reflects the quality of the original translations sent to APE.

The other baseline (**Ens8+RR**) is represented by the winning system at the WMT 2017 APE shared task (Chatterjee et al. 2017). This strong multi-component system comprises 4 different models based on the RNN architecture:

- SRC_PE a single-source model that exploits only the source sentence to generate the post-edits;
- MT_PE a single-source model that only exploits the machine translation output to generate the post-edits;
- MT+SRC_PE a multi-source model that exploits both the source sentence and the MT output to generate the post-edits;
- MT+SRC_PE_TSL another multi-source model with a task-specific loss function in order to avoid over correction.

For mixing the context vectors of the two encoders, Ens8 + RR uses a merging layer. This layer applies a linear transformation over the concatenation of the two context vectors. Chatterjee et al. (2017) compared the performance of these 4 models on the development set, and reported that MT+SRC_PE outperforms the other models. They also ensembled the two best models for each configuration to leverage all the models in a single decoder. On top of that, they also trained a re-ranker (Pal et al. 2017) to re-order the n-best hypotheses generated by this ensemble. In order to train the re-ranker, they used a set of features which are mainly based on edit distance. This set includes the number of insertions, deletions, substitutions and shifts, as well as the length ratios between MT output and APE hypotheses. It also includes precision and recall of the APE hypotheses. In Section 5, we compare our multi-source Transformer model with the ensembled model plus re-ranker (Ens8+RR). We train these models with the same

Table 2

Comparison between the Transformer-based APE and the best performing system at the WMT 2017 APE shared task.

Systems	Dev2017		Test2016		Test2017	
	TER (↓)	BLEU (↑)	TER (↓)	BLEU(↑)	TER (↓)	BLEU (↑)
MT Baseline	24.81	62.92	24.76	62.11	24.48	62.49
Ens8+RR	19.22	71.89	19.32	70.88	19.60	70.07
Transformer	18.97	72.14	18.86	71.27	19.21	70.43

settings reported in (Chatterjee et al. 2017) and, for more architecture details, we point to that paper.

4.4 System Setting

Similar to (Chatterjee et al. 2017), we initially train a generic **Transformer** model by using the $\sim 4\text{M}$ synthetic data. Then, for fine-tuning the resulting model, we try different combinations of the available data. First, in order to be comparable with (Chatterjee et al. 2017), we fine-tune the generic model on the union of the $\sim 500\text{K}$ and the in-domain training data (multiplied 20 times to give them more importance in the tuning process). Then, to analyze the contribution of each dataset, we also fine-tune the generic model separately on $\sim 500\text{K}$ instances and on the in-domain data. As we will see in Section 5, the best performance is obtained by using only the in-domain data, which puts into perspective the “the more the better” assumption commonly shared by the APE task participants. In addition to MLE loss function, when fine-tuning the generic model only on in-domain data, we use the RL-based loss functions described in Section 3.2 (**REINFORCE** and **MRT**), both alone and in combination with MLE (**MLE+REINFORCE** and **MLE+MRT**).

Our Transformer model uses word embedding with 512 dimensions. The decoder and each encoder have 4 attention layers with 512 units, 4 parallel attention heads, and a feed-forward layer with 1,024 dimensions. For training the generic model, the model parameters are updated using the Lazy Adam optimizer (Kingma and Ba 2015), with mini-batch size of 8,192. The learning rate is varied using a warm-up strategy (Vaswani et al. 2017) with warm-up steps equal to 8,000. We use a Stochastic Gradient Descent optimizer with fixed learning rate to 0.5, and mini-batch size of 2,048 tokens. The drop-out rate and the label smoothing value are set to 0.1. During decoding, we employ beam search with beam width equal to 10. For both the generic and fine-tuning steps, we continue the training for 40K steps and choose the best model checkpoints based on their performance on the development set. We use sentence-level BLEU in order to compute reward for each generated hypothesis, and for MRT we sample 5 different hypotheses. For our implementation, we use the OpenNMT-tf toolkit (Klein et al. 2017).

5. Results and Discussion

In order to have a fair comparison between our Transformer-based APE system and the best performing system in WMT 2017 APE shared task (Chatterjee et al. 2017), we fine-tuned our generic model trained on $\sim 4\text{M}$ synthetic data using the union of the $\sim 500\text{K}$

Table 3

Performance of the multi-source Transformer-based APE, fine-tuned on different types of data, on the development set.

Systems	TER (\downarrow)	BLEU (\uparrow)
MT Baseline	24.81	62.92
Generic	29.64	57.46
Union	18.97	72.14
500K	26.28	61.26
in-domain	18.49	72.23

Table 4

Performance of the Transformer-based APE fine-tuned on in-domain data using different optimization approaches on development set

Systems	TER (\downarrow)	BLEU (\uparrow)
MT Baseline	24.81	62.92
MLE	18.49	72.23
REINFORCE	22.96	66.68
MRT	22.92	66.72
MLE+REINFORCE	18.61	72.50
MLE+MRT	18.53	72.49

and in-domain data (multiplied 20). Table 2 reports the results obtained by our system (Transformer) in comparison to the system by Chatterjee et al. (2017) (Ens8+RR). From the results, it is clear that our system (Transformer) outperforms (Ens8+RR) on all the datasets. These results confirm the superiority of our system, which relies on a single multi-source Transformer model that is far less complex than the state-of-the-art multi-component architecture previously deployed (an ensemble of 8 different RNN-based models, supported by an external re-ranking component).

To analyze the contribution of each dataset ($\sim 500K$ synthetic and in-domain) to the improvement gained by fine-tuning, we separately fine-tuned the generic system on each dataset. As it is reported in Table 3, although fine-tuning only on $\sim 500K$ gains almost +15 BLEU points over the generic system, it is still lower than the MT Baseline. On the other hand, fine-tuning on in-domain data does not only outperform the MT Baseline, but it also achieves better performance than fine-tuning on the union of the two datasets. This analysis confirms that the assumption of “the more the better” in using data is not always true and, in contrast to the setting suggested by Chatterjee et al. (2017), the best performance can be obtained by fine-tuning on only in-domain data.

Furthermore, to maximize the exploitation of the in-domain data, we conducted a set of experiment using the two RL-based loss functions discussed in Section 3.2 (REINFORCE and MRT) alone and in addition to the MLE loss function. Table 4 reports the results obtained by using these loss functions on the development set. The results with both the metrics show that, although fine-tuning using only the RL-based loss

Table 5

Comparison between the multi-source Transformer-based APE and the best performing system of the WMT 2017 APE shared task on test sets of 2016 and 2017

Systems	Test2016		Test2017	
	TER (↓)	BLEU (↑)	TER (↓)	BLEU (↑)
MT Baseline	24.76	62.11	24.48	62.49
Ens8+RR	19.32	70.88	19.60	70.07
MLE	18.73	71.54	18.97	70.82
MLE+REINFORCE	18.74	71.46	18.67	71.12
MLE+MRT	18.82	71.38	18.75	71.18

functions leads to improvements over MT Baseline, their performance is still lower than the system fine-tuned using only MLE. To understand these results, we look at how the performance change during the fine-tuning process. The learning curve using RL-based loss functions is much lower than using MLE. Indeed, based on our observations during training, there is big jump in performance after few steps of fine-tuning on in-domain data using MLE, which is missing using RL-based loss functions. This is due to the fact that, in the RL loss functions, the reward is computed only once after generating the whole sequence, while MLE computes the loss after generating each word, resulting in a more coarse-grained feedback. Since, especially in industry settings, it is not cost-efficient to continue the training to reach to the maximum point using only RL-based losses, we prefer to use the RL-based loss functions together with MLE. Combining the two losses yields some improvements in terms of BLEU. Although these gains are probably no statistically significant, these results suggest that the integration of the evaluation metric in the optimization function can help the APE system.

In order to confirm our observations on the development set, we also evaluated our model fine-tuned on in-domain data using the different loss functions. Table 5 shows the results obtained on the two test sets. On the 2016 test set, the MLE loss function performs marginally better than MLE+REINFORCE and MLE+MRT. In contrast, on the 2017 test set, the addition of the RL-based loss functions to MLE (REINFORCE and MRT) outperforms, in both the cases, the system fine-tuned only using MLE.

To conclude, our experiments showed the superiority of our simple system based on a single multi-source Transformer model compared to the more complex RNN-based system by Chatterjee et al. (2017). We also demonstrated, in contrast to the previous approaches, that fine-tuning the system only on in-domain leads to better performance compared to augmenting the in-domain data with large amounts of synthetic data. Furthermore, by adding the RL-based loss functions (REINFORCE and MRT), the system can better exploit the gold in-domain data and reach better performance compared to the use of the MLE loss function alone.

6. Conclusion

In this paper, we approached the Automatic Post-Editing task focusing on effective solutions suitable for its industrial deployment. To this aim, in contrast to previous approaches, we developed a multi-source APE system that is: *i) fast to train* since it is

based on the Transformer architecture (the state of the art in MT, suitable for parallel processing) instead of RNNs, and *ii) easy to maintain* since it is based on one single model instead of multiple components. Extending our previous work (Tebbifakhr et al. 2018), we conducted a set of experiments to validate the assumption that, from the data standpoint, more data are always better for training a neural APE system. We found that this assumption is not always true. In particular, better performance can be obtained by fine-tuning the system only on gold (i.e. smaller, but higher quality) in-domain corpora. We also explored different training strategies to maximize the exploitation of small in-domain data. In particular, we used two different reinforcement learning techniques, namely REINFORCE and Minimum Risk Training, that allow us to optimize our model by considering task-specific evaluation metrics. Our experiments on the benchmark released for the WMT 2017 APE shared task show that, in a similar experimental setup, our system outperforms the best submissions to the shared task. Moreover, fine-tuning the system only on in-domain data and leveraging reinforcement learning techniques in combination with maximum likelihood leads to further improvements.

References

- Aziz, Wilker, Sheila Castilho, and Lucia Specia. 2012. PET: a tool for post-editing and assessing machine translation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3982–3987, Istanbul, Turkey, May. European Language Resources Association (ELRA).
- Bahdanau, Dzmitry, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, July.
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*, San Diego, California, USA, May.
- Béchara, Hanna, Yanjun Ma, and Josef van Genabith. 2011. Statistical post-editing for a statistical mt system. In *Proceedings of the 13th Machine Translation Summit*, pages 308–315, Xiamen, China, September.
- Béchara, Hanna, Raphaël Rubino, Yifan He, Yanjun Ma, and Josef van Genabith. 2012. An evaluation of statistical post-editing systems applied to RBMT and SMT systems. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 215–230, Mumbai, India, December. The COLING 2012 Organizing Committee.
- Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Chatterjee, Rajen, M. Amin Farajian, Matteo Negri, Marco Turchi, Ankit Srivastava, and Santanu Pal. 2017. Multi-source neural automatic post-editing: FBK’s participation in the WMT 2017 APE shared task. In *Proceedings of the Second Conference on Machine Translation*, pages 630–638, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Chatterjee, Rajen, Matteo Negri, Marco Turchi, Frédéric Blain, and Lucia Specia. 2018. Combining quality estimation and automatic post-editing to enhance machine translation output. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 26–38, Boston, MA, March. Association for Machine Translation in the Americas.
- Chatterjee, Rajen, Marion Weller, Matteo Negri, and Marco Turchi. 2015. Exploring the planet of the APEs: a comparative study of state-of-the-art methods for MT automatic post-editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 156–161, Beijing, China, July. Association for Computational Linguistics.
- Dugast, Loic, Jean Senellart, and Philipp Koehn. 2009. Statistical post editing and dictionary extraction: Systran/Edinburgh submissions for ACL-WMT2009. In *Proceedings of the Fourth*

- Workshop on Statistical Machine Translation*, pages 110–114, Athens, Greece, March. Association for Computational Linguistics.
- Gehring, Jonas, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada, July. Association for Computational Linguistics.
- Isabelle, Pierre, Cyril Goutte, and Michel Simard. 2007. Domain adaptation of mt systems through automatic post-editing. In *Proceedings of the 11th Machine Translation Summit*, pages 255–261, Copenhagen, Denmark, September.
- Junczys-Dowmunt, Marcin and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 751–758, Berlin, Germany, August. Association for Computational Linguistics.
- Junczys-Dowmunt, Marcin and Marcin Junczys-Dowmunt. 2017. The AMU-UEdin submission to the WMT 2017 shared task on automatic post-editing. In *Proceedings of the Second Conference on Machine Translation*, pages 639–646, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Kalchbrenner, Nal, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, October.
- Kingma, Diederik P. and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, California, USA, May.
- Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July. Association for Computational Linguistics.
- Knight, Kevin and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, pages 779–784, Seattle, Washington, USA, March. American Association for Artificial Intelligence.
- Lagarda, Antonio-L., Vicent Alabau, Francisco Casacuberta, Roberto Silva, and Enrique Díaz-de Liaño. 2009. Statistical post-editing of a rule-based machine translation system. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 217–220, Boulder, Colorado, June. Association for Computational Linguistics.
- Pal, Santanu, Sudip Kumar Naskar, Mihaela Vela, Qun Liu, and Josef van Genabith. 2017. Neural automatic post-editing using prior alignment and reranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 349–355, Valencia, Spain, April. Association for Computational Linguistics.
- Pal, Santanu, Sudip Kumar Naskar, Mihaela Vela, and Josef van Genabith. 2016. A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, Berlin, Germany, August. Association for Computational Linguistics.
- Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.
- Ranzato, Marc’Aurelio, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations*, San Juan, Puerto Rico, May.
- Rosa, Rudolf, David Mareček, and Ondřej Dušek. 2012. DEPFIX: A system for automatic correction of Czech MT outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, June. Association for Computational Linguistics.
- Rosa, Rudolf, David Mareček, and Aleš Tamchyna. 2013. Deepfix: Statistical post-editing of statistical machine translation using deep syntactic analysis. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 172–179, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August.

- Association for Computational Linguistics.
- Shen, Shiqi, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August. Association for Computational Linguistics.
- Simard, Michel, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, Prague, Czech Republic, June. Association for Computational Linguistics.
- Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., Montreal, Quebec, Canada, December, pages 3104–3112.
- Tebbifakhr, Amirhossein, Ruchit Agrawal, Matteo Negri, and Marco Turchi. 2018. Multi-source transformer for automatic post-editing. In *Proceedings of the Fifth Italian Conference on Computational Linguistics*, Torino, Italy, December.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., Long Beach, California, USA, December, pages 5998–6008.
- Wu, Lijun, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium, October–November. Association for Computational Linguistics.
- Zoph, Barret and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June. Association for Computational Linguistics.

