# EAT2seq: A generic framework for controlled sentence transformation without task-specific training

Tommi Gröndahl
N. Asokan

**Abstract**

We present EAT2seq: a novel method to architect automatic linguistic transformations for a number of tasks, including controlled grammatical or lexical changes, style transfer, text generation, and machine translation. Our approach consists in creating an abstract representation of a sentence's meaning and grammar, which we use as input to an encoder-decoder network trained to reproduce the original sentence. Manipulating the abstract representation allows the transformation of sentences according to user-provided parameters, both grammatically and lexically, in any combination. The same architecture can further be used for controlled text generation, and has additional promise for machine translation. This strategy holds the promise of enabling many tasks that were hitherto outside the scope of NLP techniques for want of sufficient training data. We provide empirical evidence for the effectiveness of our approach by reproducing and transforming English sentences, and evaluating the results both manually and automatically. A *single model* trained on *monolingual data* is used for all tasks without any task-specific training. For a model trained on 8.5 million sentences, we report a BLEU score of 74.45 for reproduction, and scores between 55.29 and 81.82 for back-and-forth grammatical transformations across 14 category pairs.

## 1 Introduction

Many natural language processing (NLP) tasks require representing sentences in a format that distinguishes between grammatical information and core semantic content. Traditional *formal* approaches to semantic analysis involve constructing *logical forms* (LF), which are symbolic representations presented in a theoretical metalanguage (Frege, 1879; Russell, 1905; Montague, 1970; Larson and Segal, 1995; Heim and Kratzer, 1998). Incorporating conventional LFs into NLP requires manually programmed rules that map raw text to LFs and vice versa, and resulting models cannot scale beyond these rules. LFs also lack information that would allow comparing different words, as words are simply represented by atomic symbols.

In contrast, *distributional semantics* represents words as *embeddings*, which are dense vectors computed based on the word's occurrence contexts in a corpus (Mikolov et al., 2013; Pennington et al., 2014). Unlike symbolic rules for LFs, word embeddings are not manually programmed, and allow comparing words in terms of distributional similarity, which often correlates with some aspects of semantic similarity. Dogs are more similar to cats than to houses, and this is replicated in the embedding vector distances of the words *dog*, *cat*, and *house*.

Each of these methods tackles a different task. Formal semantics models *combinatorial* (sentence- or phrase-level) information, whereas distributional semantics is concerned with *lexical* (word-level) properties. Instead of treating the approaches as inherently competitory, it is sensible to utilize the strengths of both. This task has been undertaken in a number of prior works on *natural language understanding* (Garrette et al., 2011; Lewis and Steedman, 2013; Beltagy et al., 2016).

Our focus is on another important task where abstract semantic representations are useful: *linguistic transformation*. We define this as any controlled change from a source sentence $S$ to a target sentence $T$ such that $T$ differs from $S$ only by virtue of some property $P$ specified in the transformation algorithm. The most prevalent linguistic transformation discussed in the NLP literature is *machine translation* (MT), where $P$ concerns the language in which the same (or

$$\overbrace{\text{sentence} \xrightarrow{\text{parser + EAT-rules}} \underbrace{\text{EAT-representation} \xrightarrow{\text{encoder}} \text{EAT-encoding}}_{\text{EAT2vec}} \xrightarrow{\text{decoder}} \text{sentence*}}^{\text{EAT2seq}}$$
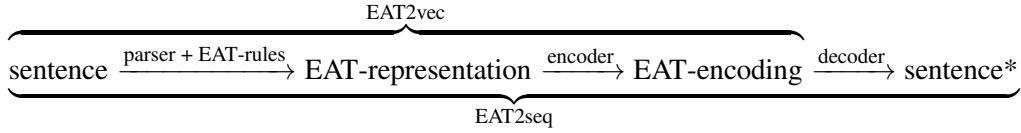
Figure 1: The EAT2vec and EAT2seq processes

maximally close) semantic content is expressed. However, transformations can also target some grammatical or lexical property within the same language. For example, in *question formation* (e.g. transforming *John saw Mary* to *Did John see Mary?*) or other *grammatical transformations*, $P$ concerns the target sentence type within the same language. Other inter-lingual transformations can involve e.g. tense, voice, or writing style.

We treat linguistic transformation as involving three parts. First, the original input text is turned into a format that maximally retains semantic information and allows straight-forward manipulation of the relevant features. Second, the transformations are applied to this representation. Finally, the output sentence is produced from the transformed input. While a large number of semantic parsing frameworks exist (Copestake et al., 2005; Bos, 2008; Berant et al., 2013; Reddy et al., 2016, 2017), text generation from abstract LFs remains highly non-trivial and typically relies either on lexical knowledge bases or parallel corpora between texts and LFs (Basile, 2015). In contrast to prior work, we tackle this task using *neural machine translation* (NMT) (Luong et al., 2015; Wu et al., 2016), and train the model on a monolingual corpus without recourse to additional resources beyond a dependency parser and pre-trained word embeddings.

In this paper, we present a general method for linguistic transformation based on using abstract sentence representations as inputs to an encoder-decoder architecture. The representations are vector sequences that approximate *Neo-Davidsonian LFs* (Higginbotham, 1985; Parsons, 1990; Schein, 1993; Pietroski, 2005) while also retaining morphosyntactic information. We denote our format as *EAT* (Event-Agent-Theme), and build the EAT-vectors from the sentence's dependency parse by using positional encoding for argument relations, pre-trained embeddings to represent words, and additional Boolean features for grammatical properties. The encoder is a recurrent neural network (RNN) that outputs a single vector representation of the entire sentence. We call this sentence embedding process *EAT2vec*. We then train a decoder RNN to reproduce the original sentence from the EAT-encoding. We use the term *EAT2seq* to refer to the entire process, as depicted in Figure 1.

We discuss four applications of EAT2seq: *grammatical and lexical transformation*, *style transfer*, *machine translation*, and *text generation*. We show that EAT2seq allows conducting them with much less training data requirements than state-of-the-art approaches. In particular, EAT2seq requires *no labeled data for training*, and allows the *same* NMT model to be used for all the tasks. Retaining the encoder and decoder weights, a single model can be applied to different tasks only by altering input pre-processing, and/or by providing additional penalty or reward to target word candidates at the decoding phase. Additionally, the model can be used to generate text from randomly selected or user-provided lexical/grammatical information, by constructing the encoder input directly from these parameters.

In addition to presenting the design of EAT2seq, we implement it by training a model on English sentences, and produce grammatical transformations between declarative/interrogative, affirmed/negated, active/passive, present/past/perfect/pluperfect, and perfective/imperfective sentences. We evaluate the transformations both manually and automatically by transforming the sentence back-and-forth, measuring the BLEU score with the original.

We summarize our contributions below.

- We present the EAT method to approximate (Neo-Davidsonian) LFs as vector sequences (Sections 3.1–3.2).

- We demonstrate how EAT-vector sequences can be used as encoder inputs to produce sentence embeddings (*EAT2vec*), and how a decoder can conversely be trained to generate a sentence from the EAT-embedding (*EAT2seq*) (Section 3.3).

- We discuss the application of EAT2seq to grammatical and/or lexical transformations, style transfer, machine translation, and text generation (Section 4).

- We design and implement a system capable of producing multiple sentence transformations, using a single EAT2seq model trained on a monolingual English corpus (Section 5).

## 2  Theoretical background

In this section we describe the theoretical basis behind our EAT-vector format. We combine morphosyntactic information with a simplified version of a common semantic formalism owing especially to Donald Davidson, later modified by others (Davidson, 1967; Castañeda, 1967; Higginbotham, 1985; Parsons, 1990; Schein, 1993; Pietroski, 2005, 2018).

### 2.1  Neo-Davidsonian logical forms

The idea that natural language expressions can be represented in a more abstract format that captures aspects of their core semantic content goes back to at least the 17th century (Wilkins, 1668). Current formalisms for expressing such theoretical constructions are typically based on philosophical and mathematical logic formulated the late 1800s and early 1900s (Frege, 1879; Russell, 1905). The most influential strand of formal semantics builds on the work of Montague (1970), who presented a general strategy for Fregean formalization of natural languages based on techniques developed by Church (1936).

A sentence is centered around its main verb. The classical Fregean/Montagovian analysis of verbs treats them as predicates that take thematic arguments (Montague, 1970; Heim and Kratzer, 1998).[1] How many thematic arguments a verb takes is treated as an inherent semantic property individually specified for each verb. For example, *run* takes one argument (an *Agent*), and *see* takes two arguments (an Agent and a *Theme*).

(1)  J = John, M = Mary

    a.  John runs
       RUN(J)

    b.  Mary sees John
       SEE(M, J)

The Fregean conception of verbs was famously challenged by Donald Davidson, who introduced an existentially quantified *event variable* as a semantic argument of all verbs (Davidson, 1967; Castañeda, 1967). An event variable is present in all verbs, and the one predicated by the main verb is existentially quantified in a declarative clause. The intuition behind this formal device is that a clause affirms (or negates) the existence of the kind of event described by the main verb.

(2)  a.  John runs
      $\exists e$ RUN($e$, J)
      'There is a running by John'

    b.  Mary sees John
      $\exists e$ SEE($e$, M, J)
      'There is a seeing of John by Mary'

Event variables allow a straight-forward analysis of various verb modifiers and arguments, such as adverbs and clausal complements, which are treated as additional predications over the event variable.

(3)  a.  John runs fast
      $\exists e$ [RUN($e$, J) $\wedge$ FAST($e$)]

---

[1]Montague semantics is formalized using Church's (1936) lambda calculus (for linguistically oriented introductions, see Partee et al. 1990; Heim and Kratzer 1998), but we use a simpler notation for readability.

b. Mary sees that John runs
$\exists e \exists e'$ [RUN($e$, J) $\wedge$ SEE($e'$, M ,$e$)]

The Davidsonian formulation retains the Fregean conception of verbs possessing an inherent semantic adicity: the fact that *run* only takes an Agent but *see* takes both an Agent and a Theme are still analyzed as inherent "lexical" properties of these verbs. An alternative to this is presented within the *Neo-Davidsonian* framework, where thematic arguments are removed from the verb, and each thematic role is allocated to a separate dyadic relation between the event variable and a thematic argument (Higginbotham, 1985; Parsons, 1990; Schein, 1993; Pietroski, 2005).

(4)  a. John runs
$\exists e$ [RUN($e$) $\wedge$ Agent($e$, J)]
b. Mary sees John
$\exists e$ [SEE($e$) $\wedge$ Agent($e$, M) $\wedge$ Theme($e$, J)]

As opposed to both the Fregean and standard Davidsonian pictures, the Neo-Davidsonian analysis assigns all verbs to the simple form P($e$): a monadic predication over an event variable. All verbs being assimilated in their semantic type, there is no longer a need to maintain argument structure information for each verb. Instead, thematic arguments can be read from the verb's *grammatical context*. Grammatical positions can be seen as providing a "spine" or "skeleton" onto which arguments can be latched, and which specify thematic relations between the arguments and the event (Hale and Keyser, 2002; Borer, 2005; Ramchand, 2008; Lohndal, 2014).

Discarding verb-particular thematic specifications thus makes it possible to analyze sentence meaning based on grammar alone. The main relevance of this theoretical idea to NLP is that it allows semantic parsing without recourse to lexical knowledge bases like FrameNet (Baker et al., 1998; Ruppenhofer et al., 2016) or PropBank (Palmer et al., 2005). Instead, at least some aspects of combinatorial semantics can be read directly from the grammatical parse. This makes the task less dependent on external resources and greatly improves its scalability.

# 3 Representing argument structure as vector sequences

In this section we describe how we map grammatical features to argument structure relations, and approximate Neo-Davidsonian LFs as sequences of fixed-size vectors.

## 3.1 Extracting argument structure from a dependency parse

*Dependency grammar* is a widely used linguistic formalism for representing grammatical properties and relations (Tesnière, 1959). A dependency parse assigns various properties to words, such as a *lemma* (inflection-neutral form), a *part-of-speech* (POS) tag, and grammatical features that provide information about the word's structural position in the sentence. We derive the argument structure representation of a sentence from such information, and use the argument structure for constructing the EAT-vectors. Here we follow prior work that has shown LFs to be reconstructable from the dependency parse alone Reddy et al. (2016, 2017). However, we cannot use such rules directly, as they are not designed for NMT architectures. Instead, we map the dependency parse to a vector sequence format that we tailor specifically for this purpose.

### 3.1.1 Thematic roles

A *dependency graph* contains information about three kinds of properties of a word: (i) its intrinsic features (e.g. POS-tag and lemma), (ii) its *head* (of which it is a *dependent*), and (iii) the nature of the relation it bears to its head. The pre-theoretical intuition behind the dependent-head relation is that some words can be seen as modifying others grammatically.

Thematic arguments of sentences are analyzed as dependents of verbs. Semantically, a verb predicates an *Event*. Active sentences can have a subject (*nsubj*) and an object (*dobj*). In line with

the standard (albeit simplified) Neo-Davidsonian approach, we map *nsubj* to *Ag(ent)*, and *dobj* to *Th(eme)*.[2] In our logical notation, we leave out existential quantification for readability.

(5)



Mary makes a sandwich

$$\text{MAKE}(e) \wedge \text{SANDWICH}(x) \wedge \text{Ag}(e, \text{M}) \wedge \text{Th}(e, x)$$

Following Hornstein and Pietroski (2009), we analyze prepositions as dyadic relations akin to thematic roles (except agentive *by*, discussed below). They relate the event or one of its arguments to another argument (see also Hale and Keyser 2002).

(6)  Mary runs with John
$$\text{RUN}(e) \wedge \text{Ag}(e, \text{M}) \wedge \text{WITH}(e, \text{J})$$

An indirect object typically corresponds to a *Recipient* argument, which is often complementary with a prepositional phrase headed by *to*.[3] We assimilate both to the prepositional construction.

(7)  Mary sings John a song / Mary sings a song to John
$$\text{SING}(e) \wedge \text{SONG}(x) \wedge \text{Ag}(e, \text{M}) \wedge \text{Th}(e, x) \wedge \text{TO}(e, \text{J})$$

Passive clause subjects (*nsubjpass*) are distinguished from active clause subjects by the dependency parse. If a direct object is present, the passive subject is a Recipient (8a). Otherwise, passive subjects are Themes (8b).

(8)  a.  John is sung a song
$$\text{SING}(e) \wedge \text{SONG}(x) \wedge \text{Th}(e, x) \wedge \text{TO}(e, \text{J})$$

   b.  A song is sung
$$\text{SING}(e) \wedge \text{SONG}(x) \wedge \text{Th}(e, x)$$

---

[2]As has been well-known at least since Perlmutter (1978), this mapping is an oversimplification, as so-called *unaccusative* verbs take Theme subjects. Consider (1a–c), where the subject corresponds to a Theme rather than an Agent, as seen in comparison to the corresponding transitive constructions (2a–c).

(1)  a.  The ice melted
   b.  The door opened
   c.  The window broke

(2)  a.  John melted the ice
   b.  Mary opened the door
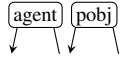   c.  Lisa broke the window

Unaccusativity has a number of grammatical correlates (see e.g. Hale and Keyser 2002; Ramchand 2008), which suggests that the respective transitive and intransitive verbs have a different grammatical status in (1a–c) and (2a–c). Hence, unaccusative verbs do not provide a genuine counter-example to the hypothesis that thematic roles are grammatically determined. The dependency grammar formalism we use is unable to detect unaccusativity, and hence our mapping assigns all subjects to the Agent role. If an additional external lexical resource was used for obtaining a list of unaccusative verbs, our EAT-format would allow representing unaccusative verbs as active verbs lacking an Agent. While this issue is highly relevant for semantic parsing, it is not an eminent problem for linguistic transformations, which is why we do not discuss it further in this paper.

[3]There are certain constructions where the complementarity fails to hold, such as (1). (We denote ungrammaticality by "*".)

(1)  a.  John makes Mary a pizza
   b.  *John makes a pizza to Mary

Such exceptions indicate that the grammatical relation between datives and *to*-constructions is more complex than simple equivalence. EAT2seq automatically learns the distribution between the constructions with respect to particular verbs, and will never see ungrammatical examples like (1b) during training.

Agents introduced in passive constructions via the preposition *by* are dependents of this preposition, which relates to the verb via the relation marked *agent* in the dependency graph.

(9)

$$\text{agent} \quad \text{pobj}$$

A sandwich is eaten by Mary

$$\text{EAT}(e) \land \text{SANDWICH}(x) \land \text{Ag}(e, \text{M}) \land \text{Th}(e, x)$$

Our relatively simple argument structure representation can be used for arbitrarily complex sentences, since it allows for recursive application of arguments inside arguments. An example of this is the clausal Theme of (10) with its own thematic roles.

(10) John sees that Mary walks
$$\text{SEE}(e) \land \text{WALK}(e') \land \text{Ag}(e, \text{J}) \land \text{Th}(e, e') \land \text{Ag}(e', \text{M})$$

### 3.1.2 Modifiers

All lexical[4] words can be modified by additional elements. We recognize the following modifiers: *prepositional phrases, possessors, numerals, determiners, adjectives, adverbs, modal auxiliaries,* and *compound elements*.

As mentioned in Section 3.1.1, we represent prepositions as dyadic relations akin to thematic roles. Prepositional phrases can modify an event or any of its arguments.

(11) A man in a suit runs on the roof
$$\text{RUN}(e) \land \text{MAN}(x) \land \text{SUIT}(y) \land \text{ROOF}(z) \land \text{Ag}(e, x) \land \text{IN}(x, y) \land \text{ON}(e, z)$$

Possessors are nouns that modify other nouns via the ownership/possession relation.

(12) Mary's dog runs
$$\text{RUN}(e) \land \text{DOG}(x) \land \text{Ag}(e, x) \land \text{POSS}(\text{M}, x)$$

Numerals or adjectives modify nouns, and adverbs modify verbs. We treat all of them as simple monadic predicates.[5]

(13) Three brown dogs run fast
$$\text{RUN}(e) \land \text{FAST}(e) \land \text{DOG}(x) \land 3(x) \land \text{BROWN}(x) \land \text{Ag}(e, x)$$

Some modifiers are more complex in their interpretation. To maximize simplicity in implementation, we gloss over some such details at the expense of theoretical rigor. In particular, we treat compounds elements, modal auxiliaries and degree adverbials as if they were simple modifiers.

(14) An ant eater can run very fast
$$\text{RUN}(e) \land \text{CAN}(e) \land \text{FAST}(e) \land \text{VERY}(e) \land \text{EATER}(x) \land \text{ANT}(x) \land \text{Ag}(e, x)$$

More discussion of such *non-entailing modifiers* is provided in Appendix A.

## 3.2 Representing argument structure as vector sequences

In this section we describe how we translate an argument structure representation into a sequence of vectors that approximate parts of a Neo-Davidsonian LF with appended grammatical features. We use the term *EAT* (Event-Agent-Theme) for the sequence format. While the number of vectors in EATs is unlimited in principle, a fixed size for each vector is required in order for the sequence to function as an input for an RNN. The EAT-vectors replace word embeddings in standard neural machine translation (NMT) encoder architectures.

---

[4]Lexical words comprise nouns, verbs, adjectives, and adverbs. Some analyses also include prepositions, although this is contested. For a theoretical overview and extensive analysis, see Baker (2003).

[5]Treating numerals as predicates means that we cannot interpret the argument variables as simply denoting individuals. For discussion on *plural variables* see Boolos (1984); Schein (1993); Pietroski (2003).

### 3.2.1 Arguments and modifiers

We represent thematic relations as word triplets, with positional encoding for roles. To begin with, a basic Event-Agent-Theme relation translates to the triplet of the respective words in this order.

(15) $<W_1, W_2, W_3> \Leftrightarrow W_1(e) \wedge W_2(x) \wedge W_3(y) \wedge \text{Ag}(e, x) \wedge \text{Th}(e, y)$

The triplet representation is most intuitive for simple transitive sentences like (16).

(16) A dog sees a cat
     $<$see, dog, cat$>$

Verbs that lack either the Agent or Theme argument include a special empty token in this position, denoted here as $\varnothing$.

(17) a. A dog walks
       $<$walk, dog, $\varnothing>$
    b. A cat was seen
       $<$see, $\varnothing$, cat $>$

The basic triplet format also extends to basic (i.e. non-prepositional) modifiers, which are simply additional predications of the event or its arguments. For example, the sentence *A brown dog walks* entails that (i) something that is a dog walks, and (ii) something that is brown walks. We can thus represent it as a sequence of Event-Agent relations.

(18) A brown dog walks
     $<$walk, dog, $\varnothing> <$walk, brown, $\varnothing>$

However, to improve model performance by avoiding redundancy, we use the empty token for repeated elements.

(19) A brown dog walks
     $<$walk, dog, $\varnothing> <\varnothing$, brown, $\varnothing>$

Possessors are like other basic modifiers, but contain an additional possessive feature.

(20) Mary's dog walks
     $<$walk, dog, $\varnothing> <\varnothing$, Mary$_{\text{Poss}}$, $\varnothing>$

We analyse prepositions as dyadic relations (3.1.1–3.1.2). A straight-forward solution would thus be to assimilate them to Events, and their arguments to thematic roles. However, this involves repetition, as the preposition's "Agent" will also appear elsewhere in the sequence.

(21) A dog in a house sees a cat on a roof
     $<$see, dog, cat$> <$in, dog, house$> <$on, cat, roof$>$

Simply removing the "Agent" would also delete the information concerning the argument the preposition modifies. To avoid this while still preventing repetition, we denote the argument as the Event/Agent/Theme position, allocating this feature to the preposition.

(22) A dog in a house sees a cat on a roof
     $<$see, dog, cat$> <$in$_{\text{Ag}}$, $\varnothing$, house$> <$on$_{\text{Th}}$, $\varnothing$, roof$>$

### 3.2.2 Grammatical features

In addition to thematic structure, we append the EAT-tuples with indicators of grammatical properties, which are relevant during the decoding phase of EAT2seq. All grammatical features are derived from the dependency parse. The features and their possible values are listed below.

**Verbal/clausal features:**

*Force*: declarative/imperative/question

*Truth*: true/false[6]

*Voice*: active/passive

*Tense*: past/present/perfect/pluperfect

*Aspect*: perfective/imperfective

**Nominal features:**

*Number*: singular/plural

*Definiteness*: definite/indefinite

*Possessive*: possessive/non-possessive

**Adjectival/adverbial features:**

*Comparison class*: comparative/superlative

When the Event position is occupied by a preposition, we additionally mark it as modifying an Event, Agent, or Theme.

All verbs are either active or passive, and hence both features being 0 indicates that the Event position is occupied by a non-verb, i.e. a preposition, a verb modifier (e.g. an adverb), or the empty token. Tense is represented by three binary features: present, past, and perfect. We refer to the present perfect as "perfect" and the past perfect as "pluperfect". If a verb lacks all tense features, it bears the infinitival inflection.

### 3.2.3 Relative pronouns

We represent relative pronouns as regular arguments (*that*, *which*, *who* etc.), but add additional features that associate them with other elements in the EAT-sequence. We include the following four binary features to the representation:

R1: Agent is a relative pronoun modifying another Agent

R2: Agent is a relative pronoun modifying a Theme

R3: Theme is a relative pronoun modifying an Agent

R4: Theme is a relative pronoun modifying another Theme

Examples of each are provided in (23).

(23)  a.  John, who likes Lisa, sees Mary
          <TRUE, see, John, Mary> <R1, TRUE, like, who, Lisa>

     b.  John sees Mary, who likes Lisa
          <TRUE, see, John, Mary> <R2, TRUE, like, who, Lisa>

     c.  John, who Lisa likes, sees Mary
          <TRUE, see, John, Mary> <R3, TRUE, like, Lisa, who>

---

[6]Even though we call the token "Truth" for notational convenience, it does *not* represent the truth value of the tuple as such. Rather, it is simply a marker for whether the Event is negated ("false") or affirmed ("true").

| Feature class | Index | Feature meaning |
|---|---|---|
| Force | 1 | Question |
| | 2 | Imperative |
| Truth | 3 | True |
| Voice | 4 | Active |
| | 5 | Passive |
| Tense | 6 | Present |
| | 7 | Past |
| | 8 | Perfect |
| Aspect | 9 | Imperfective |
| Preposition | 10 | Preposition modifying Event |
| | 11 | Preposition modifying Agent |
| | 12 | Preposition modifying Theme |
| Noun | 13 | Agent is possessive |
| | 14 | Agent is plural |
| | 15 | Agent is definite |
| | 16 | Agent is an Agent-RP |
| | 17 | Agent is a Theme-RP |
| | 18 | Theme is possessive |
| | 19 | Theme is plural |
| | 20 | Theme is definite |
| | 21 | Theme is an Agent-RP |
| | 22 | Theme is a Theme-RP |
| Adjective/adverb | 23 | Event is comparative |
| | 24 | Event is superlative |
| | 25 | Agent is comparative |
| | 26 | Agent is superlative |
| | 27 | Theme is comparative |
| | 28 | Theme is superlative |

Table 1: Boolean grammatical features in the EAT-vector format (RP = relative pronoun)

    d. John sees Mary, who Lisa likes
        <TRUE, see, John, Mary> <R4, TRUE, like, Lisa, who>

If a relative pronoun lacks all the features R1–R4, it modifies an Event, as in (24).

(24)   John runs, which Mary sees
        <TRUE, run, John, $\varnothing$> <TRUE, see, Mary, which>

### 3.2.4 Vector format

We transform each EAT-tuple in the sequence into a vector using Boolean encoding for grammatical and negation features, and pre-trained embeddings to represent words. The grammatical features are listed in Table 1. The length of each vector is thus $28 + 3S$, where $S$ is the embedding length. The first 28 Boolean components specify grammatical features, and the rest is derived by concatenating the embeddings of the Event, Agent, and Theme, in this order. The empty token $\varnothing$ corresponds to the zero embedding. We begin the EAT-sequence of each sentence with the thematic specification of the main clause, the root verb occupying the Event position. This makes it possible to transform features of the main clause by altering the features of the first EAT-vector (Section 4.1).

### 3.3 Pipeline

We use the sequence of $28 + 3S$ -sized EAT-vectors as input for an encoder RNN, replacing word embeddings in standard NMT architectures. The encoder outputs a single vector representing the entire sentence. We then give the EAT-encoding as an input to a decoder RNN, which we train to reproduce the original sentence. The entire process can be described with the pipeline depicted in Figure 2, where part (i) is done by an external parser, part (ii) is rule-based as described in Section 3, and part (iii) is learned via NMT. We denote the stage terminating at the EAT-encoding as *EAT2vec*, and the entire process as *EAT2seq*.

$$\overbrace{\text{source} \xrightarrow{\text{parser}} \text{parse}}^{(i)} \overbrace{\xrightarrow{\text{EAT-rules}} \text{EAT-representation}}^{(ii)} \overbrace{\xrightarrow{\text{encoder}} \text{EAT-encoding} \xrightarrow{\text{decoder}} \text{target}}^{(iii)}$$
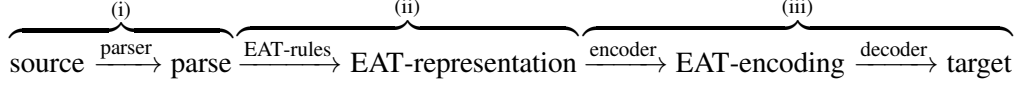
Figure 2: The EAT2seq pipeline

## 4  Applications

We now turn to how EAT2seq can be utilized for controlled text transformation both within a language and between languages, as well as generating text directly from lexical and grammatical information. We focus on four tasks: linguistic transformation (both grammatical and lexical), style transfer, dictionary-based machine translation, and text generation. We emphasize that *all* the tasks can be performed with a single model trained only on monolingual English sentences.

### 4.1  Linguistic transformation

In this section we describe how EAT2seq allows user-controllable changes to grammatical or lexical content of an input sentence. As the first EAT-vector in the sequence always corresponds to the main clause (Section 3.2.4), the alterations can be restricted here.

#### 4.1.1  Grammatical transformation

By *grammatical transformation* we mean a controlled non-lexical change to a text that either has little to no effect on the semantic content, or only has a systematic grammatical effect. By *non-lexical* we mean that grammatical changes should not alter lexical ("content") words but rather the ways in which words are arranged or expressed in surface-level grammar. Further, we require the changes to be in the user's *control*, i.e. predictable from parametric choices the user can make. Automatic grammatical transformations can be used in many applications, involving e.g. language pedagogy, style transfer (see Section 4.2), or question answering.

Given that the EAT-vector sequence makes a distinction between grammatical markers (28 Boolean features) and propositional content (Event-Agent-Theme), controlled transformation of the former are possible without affecting the latter. This process takes place at the level of EAT-vectors prior to encoding, and is shown in Figure 3.

As an example, consider the EATs generated from the sentence *John saw Mary*, and the corresponding question *Did John see Mary?*. (We omit most grammatical features in our exposition for readability.)

(25)   a.   John saw Mary
        <DECLARATIVE, TRUE, ACTIVE, PAST, see, John, Mary>
    b.   Did John see Mary?
        <QUESTION, TRUE, ACTIVE, PAST, see, John, Mary>

The question-EAT can easily be created from the declarative-EAT by changing the first feature from DECLARATIVE to QUESTION, and vice versa. Thus, provided that the decoder can reproduce both the declarative and the question from the respective EATs, adding very simple

$$\text{source sentence} \xrightarrow{\text{parser + EAT-rules}} \text{EAT}_{\text{orig}} \xrightarrow{\text{transformations}} \text{EAT}_{\text{transf}} \xrightarrow{\text{encoder + decoder}} \text{target sentence}$$

Figure 3: EAT2seq-based linguistic transformation

pre-processing rules prior to encoding allows using EAT2seq for transforming a declarative into a question and back.

The same principle applies to all grammatical features that are relevant for the sentence's main verb: force, negation, voice, tense, and aspect. Hence, the approach allows generating passive counterparts of active sentences, negating a sentence, changing tense, etc. Additionally, such changes can be enacted *in any combination* without increasing the speed or difficulty of the process. The only requirement for success is that the decoder can successfully create the target sentences, and hence that the training corpus contains enough example sentences in the relevant class. This requirement is independent of the amount of transformations applied at the EAT-level.

Its potential use for grammatical transformations demonstrates the virtues of EAT2seq especially well. In particular, it combines the benefits of rule-based and data-based NLP without introducing the main drawbacks of either method. Rule-based approaches are notoriously unscalable and difficult to program, but they require no training and allow precise user control over the output. Data-based methods scale better and have a generally superior performance in most NLP tasks, but are data-hungry and grant much less user control over the process.

EAT2seq uses rule-based transformations to attain the EAT-vector sequence from (parsed) text, and at this stage it allows human-readable manipulations to the input to affect the output in predictable ways. Transformation rules on the EAT-level are maximally simple, only changing between Boolean feature values. Additionally, no parallel corpus or labeled training set is needed to train the model to enact such transformations, as the changes take place during input pre-processing prior to encoding. After this, an encoder-decoder network produces the output, freeing the user from having to program the transformations from EATs back to English. We thus retain the benefits of rule-based approaches in allowing user control of the output and minimizing training requirements, while also achieving the state-of-the art performance of data-based NMT in mapping EATs to English via an encoder-decoder network.

### 4.1.2 Lexical transformation

As the mirror-images of grammatical transformations, we take *lexical transformations* to alter some content word(s) in the sentence, leaving grammatical information intact. These are as straight-forward to produce as grammatical changes. In the EAT-representation, a strict distinction is drawn between grammatical features and content words that participate in thematic relations. Analogically to changing a Boolean grammatical parameter between the values 0 and 1, we can change one or more of the content words. Since these are represented as lemmas (prior to being replaced by pre-trained word embeddings), such replacement is independent of inflectional considerations.

The user can specify which of the three words present in a EAT-vector she wants to change, and can also *remove* a thematic argument. This is a special type of lexical change, where the argument is replaced with the empty token.[7] As an example task, consider the generation of sentences entailed by the original. One way to achieve this is to replace an original word by a *hypernym*, i.e. a word with a larger extension subsuming that of the original. Information about hypernymy is provided by lexical knowledge bases such as WordNet Miller (1995). This allows selecting a random hypernym from such a knowledge base, and replacing the original word with it. An example of such a transformation is *John saw a dog* ⇒ *John saw an animal*, the schematic derivation of which via EAT2seq is provided in (26) (ignoring tense).

(26)   $\langle\text{see, John, dog}\rangle \xrightarrow{\text{Theme: hypernym}} \langle\text{see, John, animal}\rangle \xrightarrow{\text{decode}}$ John saw an animal

---

[7]Argument removal additionally involves removing all modifiers of the argument. EAT-representations contain enough information to allow this, but we gloss over details here.

In negated sentences and certain other cases (e.g. universal quantification), replacing a word with its *hyponym* (i.e. the opposite of a hypernym) results in entailment: *John did not see an animal ⇒ John did not see a dog*. Finding the correct entailment contexts for hypernym or hyponym replacement is nontrivial, but is greatly simplified by using EAT-representations. For instance, detecting negation is straight-forward from the EAT, as it is among the Boolean grammatical features.

Alternatively, we can also generate sentences that entail the original, thus providing more exact information. If the sentence is also transformed into a question and a first-person agent changed into the second person, this method provides a simple question-transformation algorithm that inquires more specific information from the user, as in transforming (27a) to (27b) or (27c), where *dentist* and *orthodontist* are alternative hyponyms of *doctor*. Such a system could e.g. be used for eliciting user replies in a customer service setting, or incorporated into a chat-bot.

(27)   a.  I want to see a doctor.

       b.  Do you want to see a dentist?

       c.  Do you want to see an orthodontist?

In summary, EAT2seq allows lexical as well as grammatical transformations of the input sentence. Especially when applied in combination, such transformations can have many potential uses, including (but not limited to) entailment production and question/answer generation.

## 4.2   Style transfer

Authors often have a unique writing style. The claim that authors leave a computationally tractable stylistic trace is known as the "human stylome hypothesis" (van Halteren et al., 2005). The field of *stylometry* focuses on text classification based on surface-level stylistic markers. In addition to author identification, it can be used for profiling author properties such as age, gender, religion, political stance, etc. Multiple studies have shown stylometry to succeed in author identification with a high accuracy (Zheng et al., 2006; Narayanan et al., 2012; Castro and Lindauer, 2013).

While author identification has many useful applications (e.g. Juola 2013), it also poses a privacy threat by allowing deanonymization against the author's will (Brennan and Greenstadt, 2009; Brennan et al., 2012). To combat such *deanonymization attacks* (Narayanan et al., 2012), the author can modify the text to retain most of its content while changing features relevant for stylometric classification. We call this *style transfer*. In this section, we describe how EAT2seq can be used for style transfer in a more controlled manner than alternative approaches suggested in prior literature.

The simplest approach to using EAT2seq for style transfer involves training the encoder-decoder model on text written in a particular style, and applying this model to a source text written in another style. As a simple example, consider the distribution of negations as either separate words (*not*) or in contracted form (*n't*). Since these are assimilated in the EAT-representation, which one the decoder will produce depends on their distribution in the target corpus. Suppose, for instance, that the target corpus most often marks negation as *not* following *could*, but as *n't* following *do*. Consequently, the decoder would prefer *could not* and *don't* to the alternative forms *couldn't* and *do not*.

Alternatively, it is also possible to include author properties as meta-features to the EAT-representation itself, akin to the Boolean grammatical features. The EAT2seq model can be trained on data from authors $A_1, ..., A_n$, and each EAT-vector can be appended with $n$ Boolean features indicating the author. For example, if for some $i, j \in \{1, ..., n\}$, $A_i$ prefers *do not* and $A_j$ prefers *don't*, the author feature will impact the decoder's choice of a negation marker following *do*. Using author features allows the model to be trained with more data, as more author corpora can be included instead of only one.

The approach described so far requires style-specific training. It continues the line of recent studies where a latent sentence representation is mapped onto different target styles via either separate decoders or a single decoder with style features as inputs (Shen et al., 2017; Fu et al.,

2018; Shetty et al., 2018). However, EAT2seq also allows delegating the whole style transfer process to the pre-trained decoder, without requiring any style labels at training. In the beam search stage during decoding (see Section 5.1), candidate words can be punished or rewarded based on the frequency of word- and/or character n-grams in *arbitrary* source and target corpora provided by the user. For instance, if *not* is more common than *n't* in the target corpus, its probability would be increased and the probability of *n't* decreased. This approach would constitute a general style transfer method capable of imitating in principle any target corpus with a single model.

Whether the style-specificity is achieved by training the model on specific styles or by adjusting decoder probabilities in a pre-trained model, the basic idea behind both is that EAT-representations abstract away from many stylistic aspects that are then constructed by the decoder. Therefore, adjusting target word probabilities during decoding can be used to control target style. The pipeline is described in Figure 4.

$$\text{source sentence} \xrightarrow{\text{EAT2vec}} \text{EAT-encoding} \xrightarrow{\text{target style decoder}} \text{target sentence}$$

Figure 4: EAT2seq-based style transfer

We consider using EAT2seq for style transfer to be an important aspect of future work, focusing in particular on the prospects of using the same model for any target without additional training.

## 4.3 Machine translation

In addition to translating between different styles within the same language, EAT2seq also has potential to be used for MT between a source language (SL) and a target language (TL), using rules for obtaining SL-EATs, and a lexical SL-TL dictionary, and a pre-trained EAT2seq model transforming EATs into the TL. While this approach is limited to cases where word-to-word translations are available (as opposed to word-to-phrase, phrase-to-word, or phrase-to-phrase), we believe it provides a valuable addition to the recent field of unsupervised MT (Artetxe et al., 2017; Lample et al., 2017, 2018).

The idea of a language-independent representation mediating between the SL and TL goes back to the beginning of automatic translation as a theoretical suggestion (e.g. Wilkins 1668). Unsurprisingly, such *interlingual* translation has remained a theoretical idea without plausible implementation, although some small-scale attempts exist (e.g. Dorr 1993). EAT2seq allows the basic idea to be partially re-incorporated within data-based MT, without sacrificing the scalability of state-of-the-art NMT. This possibility is built around the assumption that there is a distinction between combinatory semantics and word meaning, the former but not (necessarily) the latter being shared between multiple languages.

Arguably, combinatorial differences between languages arise not from the combinatorial relations themselves (such as Event-Agent-Theme), but how these map to grammatical relations in complex but nevertheless tractable ways. These mapping principles allow language-specific EAT-rules to be applied to a grammatical parse, as described in Section 3 for English. Once the combinatorial relations are uncovered, the EAT-sequence can be built in a manner that maximizes uniformity across different languages and writing styles. Hence, the EAT-encoding functions as an approximation of an interlingual representation concerning combinatorial properties. If an additional dictionary is available for lexical mapping between the SL and TL, EAT2seq can be used as a partially interlingual translation scheme.

The first attempts at MT were dictionary-based, with famously underwhelming results (Hutchins, 2010). There were a number of reasons for this, not the least of which was the difficulty of achieving grammatical TL expressions using only dictionary information on the level of uninflected lemma forms. In rule-based MT, mitigating this problem was attempted within the *transfer* approach. Here, the SL was first transformed into a more abstract grammatical representation, which was then translated into a similar abstract TL representation, from which the final TL expression was produced. The transfer method occupied a middle ground between direct

translation rules from the SL to the TL (dictionary methods) and the theoretically intriguing but practically unrealistic interlingua method.

Analogically to the transfer approach to MT, EAT2vec can be used as an intermediate stage between the SL and TL. However, unlike in the transfer method, all grammatical transformations from the EAT-encoding to the TL are learnt from the monolingual target corpus during training. Since the EAT-sequences are language-independent with the exception of lexical content,[8] it follows that translation to the TL is possible by (i) making an EAT-sequence of the SL-sentence, (ii) translating lexical material (i.e. Event/Agent/Theme positions) with a dictionary, and (iii) giving the dictionary-translated EAT-sequence as an input to the encoder-decoder network producing a TL sentence. This pipeline is depicted in Figure 5.

$$\text{source sentence} \xrightarrow{\text{parser + EAT-rules}} \text{EAT}_{SL} \xrightarrow{\text{SL-TL-dictionary}} \text{EAT}_{TL} \xrightarrow{\text{encoder + decoder}} \text{target sentence}$$
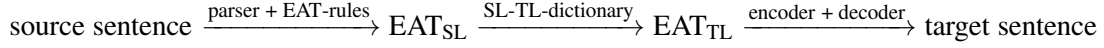
Figure 5: EAT2seq-based MT

As an example, consider translating the German sentence (28a) to (28b) in English.

(28)  a.  John hat    einen        Hund gekauft
          John AUX INDEF.ACC dog    buy.PTCP
      b.  John bought a dog

As can be seen from the gloss, there are a number of grammatical differences between the original sentence and its translation: the word orders differ, (28a) contains an auxiliary which does not correspond to anything visible in (28b), and (28a) contains a number of German-specific inflectional considerations such the gender of *Hund*, the accusative case of the indefinite article, and the participial inflection of the verb.

In translating from (28a) to (28b) using EAT2seq, German-particular inflectional features would first be stripped off the German EAT, which would then be changed into an English EAT by lexical mapping, which in turn would be transformed to English by the encoder-decoder network. In this MT architecture, no direct mappings exist between language-specific inflectional rules of the SL and TL. Grammatical features of the source sentence are handled by the SL parser and EAT-generation, and grammatical features of the source sentence are handled by the TL-decoder. Both are independent of the translation task.

Of course, the method is limited to those cases where a word-to-word mapping can be achieved; i.e. translation does not take place between phrases of different sizes. Other limits to dictionary-based MT also remain, concerning in particular word sense disambiguation in different contexts. Realistically, we believe that EAT2seq can provide important assistance in the growing field of unsupervised MT, but would face challenges as a stand-alone solution. Applying EAT2seq to MT constitutes an important aspect of future work.

We have thus taken a step closer to reconsidering the feasibility of dictionary-based translation in MT. Here, language-specific grammatical aspects of the SL and TL are never directly mapped to each other, but instead processed separately by the SL-parser and the EAT-TL decoder. This method would therefore greatly reduce the size requirements on the parallel corpus, as the amount of word-to-word mappings is a small fraction of the amount of sentence-to-sentence mappings. Unsupervised methods for inferring lexical mappings in the absence of parallel corpora have also been developed (Conneau et al., 2018; Artetxe et al., 2018; Alvarez-Melis and Jaakkola, 2018). A crucial benefit of the proposed MT-architecture is that a single EAT-TL decoder can be used for translating *any SL* to the same TL, provided the SL-EAT mapping can be achieved (i.e. a SL-parser and a SL-TL dictionary are available). This is in line with our general argument that a single EAT2seq model can have a number of different uses without requiring additional training.

---

[8]The claim of language-neutrality is, of course, an idealization. The grammatical markers we use are based on English, and should be adjusted if EAT2seq was used for translating between typologically different languages. Nevertheless, we maintain that Event, Agent and Theme are plausibly universal semantic categories, or at least not specific to English.

In summary, despite the difficulties involved in dictionary-based lexical translation, we believe EAT2seq can have a role in improving unsupervised MT.

## 4.4 Text generation

Finally, we discuss an application of EAT2seq where the EAT-representation is built without parsing a source sentence. The EAT-vector can be constructed directly from lexical and grammatical information about the thematic roles and their grammatical properties. As verbs and their thematic arguments can also be randomly chosen from a corpus, EAT2seq can be used for generating random sentences with possible user control of *any* lexical or grammatical feature(s). Thus, the same EAT2seq architecture that makes possible controlled text transformation also allows *controlled text generation*. The pipeline is shown in Figure 6.

$$\text{input words} + \text{grammatical parameters} \xrightarrow{\text{EAT-construction}} \text{EAT} \xrightarrow{\text{encoder + decoder}} \text{target sentence}$$
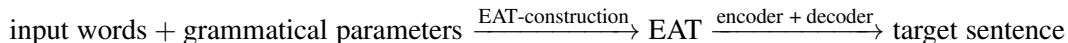
Figure 6: EAT2seq-based text generation

Control over lexical or grammatical aspects of generated text is needed for *targeted generation*, where the produced text needs to fulfil certain conditions while also retaining sufficient randomness of content. A recent example of such a task is the generation of contextually appropriate artificial restaurant reviews by Juuti et al. (2018). Prior approaches to controlled generation have required the control parameters to be set in a model-specific way, and to be present in training data labels (Hu et al., 2017; Rao and Tetreault, 2018). In contrast, EAT2seq allows the user to control in principle *any* grammatical or lexical aspect, in *any combination*, without using labeled training data.

To demonstrate the use of such a text generation system, we implemented a small toy generator which chooses a random Event from a list of transitive verbs, and a random Agent and Theme from a list of common nouns.

Verbs: *see, hear, chase, love, like, hate, call*

Nouns: *dog, cat, man, woman, boy, girl, person*

The EAT consists of a single vector with all Boolean grammatical features as zero except the truth value (True), tense (Past), and voice (Active). This functions as the input for the encoder-decoder network we implemented, as described in Section 5.1. Some example sentences produced by the generator are presented in (29).

(29)   a.   A person called a dog
      b.   A man chased a girl.
      c.   A boy heard a cat.
      d.   A cat liked a person

As the example sentences indicate, the generator restricts the grammar while choosing the arguments randomly. This toy implementation demonstrates that EAT2seq can be used for text generation as well as transformation. The generation method allows far more user control than prior approaches, without requiring task-specific training. This additional use case adds to the benefits of EAT2seq as a general NLP tool, considering especially applications such as chat-bots, as also discussed in Section 4.1.2.

## 5   Implementation and experimental results

In this section we describe the technical details of our EAT2seq implementation, and discuss the experimental results we received from the text reproduction and grammatical transformation tasks. The experiments are intended as indicative evidence of the power and viability of EAT2seq, rather

than as comprehensive evaluations of all applications described in Section 4. We defer systematic evaluations of further applications to future work.

## 5.1 Implementation

In this section we review the technical details of our EAT2seq implementation in terms of the basic architecture (5.1.1), as well as additional algorithms we used to improve model performance (5.1.2–5.1.3).

### 5.1.1 Model architecture

We implemented EAT2seq on Python 3, using SpaCy[9] for dependency parsing, and Pytorch for programming the encoder and decoder RNNs. Both the encoder and decoder were two-layer LSTM networks (Hochreiter and Schmidhuber, 1997) with 600 hidden units in each layer. As pre-trained word embeddings we used 300-dimensional GloVe vectors (Pennington et al., 2014) trained on a Common Crawl corpus.[10] Our EAT-vectors thus had 928 components overall.

In the forward pass, we initialized the decoder hidden state with the final encoder output. We further applied an *attention* mechanism (Bahdanau et al., 2014) by using intermediate encoder outputs as additional decoder inputs. Our attention length was 9. Since the maximum EAT-sequence length in the training set was 10, all intermediate outputs could affect the decoding.

During training we applied a batch size of 128, the negative log likelihood loss function, and a dropout probability of 0.1. Using the Adam optimizer (Kingma and Ba, 2014), we began with a learning rate of 0.001 and reduced it to 0.0001 after epoch 5 to increase training speed. To measure overfitting we used an additional validation set of 50000 sentences. After epoch 7 reducing the learning rate no longer decreased validation loss, and we used the weights from this epoch in our tests. Our final training loss was 0.68 and the validation loss 0.78.

For training the networks, we used 8.5 million English sentences derived from multiple corpora, listed below.

- *Stanford parallel corpora*: English-German and English-Czech machine translation corpora (Luong et al., 2015; Luong and Manning, 2016)[11]

- *OpenSubtitles 2018*: a subtitle corpus[12]

- *Tatoeba*: a multilingual machine translation corpus[13]

- *SNLI*: a natural language inference corpus (Bowman et al., 2015)[14]

- *SICK*: a natural language corpus for compositional inference (Marelli et al., 2014)[15]

- *Aristo-mini (December 2016 release)*: a corpus of answers to scientific questions[16]

- Example sentences from *WordNet* (Miller, 1995)[17]

To simplify the task, we only used sentences that had at most 20 words. We also discarded sentences that produced no EAT or contained no main verb. We optimized the training set by making EATs from all sentences in the corpora (after size filtering, resulting in 20 million sentences altogether), and selecting the training set by distributing the grammatical classes as equally

---

[9] https://spacy.io/

[10] http://commoncrawl.org/. The GloVe vectors are provided by Spacy's large English model.

[11] https://nlp.stanford.edu/projects/nmt/

[12] http://opus.nlpl.eu/OpenSubtitles2018.php

[13] https://tatoeba.org

[14] https://nlp.stanford.edu/projects/snli/

[15] http://clic.cimec.unitn.it/composes/sick.html

[16] https://www.kaggle.com/allenai/aristo-mini-corpus

[17] http://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html

as possible. The motivation for this was to avoid class imbalance, which can have a negative effect on transformation success (as discussed in Section 5.2.4).

At the test phase, we applied additional means to increase the likelihood of reaching the correct target. Experimentation on a validation set (outside both the training and test sets) indicated that these measures improved model performance. We describe them in 5.1.2–5.1.3, and adopted all in the experiments reported in Section 5.

### 5.1.2 Placeholder replacement

To make the model scale better to novel data, we replaced proper names, numbers, and unknown words with random placeholders from small lists of words known to the model. The placeholders were then changed back to the original words after decoding. Placeholder replacement allows the model to transform sentences with arbitrary names or numbers, as well as to retain words for which no pre-trained embedding exists.

The placeholders for names were taken from a list of 15 common first names recognized by the pre-trained GloVe embeddings.[18] In the training set, we had randomly replaced all names with these, making them the only names recognized by the model. We did the same with plural numbers, using random digits between 2 and 9 as placeholders. We further contracted complex names and numbers (i.e. name/number compounds) into only one placeholder, simplifying the training sentences. At the test phase, each name or number was first replaced with a random placeholder, and then mapped back to the original after decoding. Again, complex names and numbers were treated as single chunks replaced with only one placeholder. We only used placeholders which did not appear elsewhere in the original sentence.

We further extended placeholder replacement to words that were unknown to the GloVe embedding matrix. At the test phase, we replaced these with placeholders taken from small lists for dedicated lexical classes, and brought the original words back at post-processing. We used the following placeholder words, chosen manually:

- Intransitive: *walk, sing, eat, drink, sit*

- Transitive: *make, do, prepare, sing, write*

- Count: *woman, man, girl, boy, dog*

- Mass: *stuff, water, air, fire, food*

To allow both transitive and ditransitive verbs to be replaced without additional complications, we chose the transitive verb placeholders to allow but not require an indirect object. We used the mass noun placeholders for unknown words in other POS categories.

We stored all possible inflections of each placeholder, and allowed the replacement only if none of the placeholder's forms appeared elsewhere in the original sentence. After the decoding, inflected placeholders were first mapped back to their lemmas, and thereby back to the original unknown words.

### 5.1.3 Punishment during beam search

We trained the decoder with greedy search, but used beam search ($k = 10$) in the test phase. This allowed us to provide additional punishment to inappropriate candidate words. We punished the unknown token, words repeated more times than in the original sentence, the same word appearing twice in a row (unless the original sentence contained this), and name/number placeholders outside the ones used for replacement. To ensure these were never chosen, we applied a punishment of 10000 to their log-probabilities.

We further noticed that the model sometimes struggled with passive sentences containing an Agent, mistakenly adding this before the Theme. To avoid this, we punished a passive Agent's

---

[18]The name placeholders were: *John, Mary, Bob, Alice, Lisa, Tom, Harry, Anna, James, Jennifer, Richard, Charles, Thomas, George,* and *Linda.*

log-probability with 10 if the Theme had not already been added. The algorithm further covered Agent and Theme modifiers in addition to the head words. However, to avoid overly excessive punishments, we only applied this if the Theme was present in *some* candidate. Effectively, this punishment required the model to prefer candidates where the Theme precedes the Agent, when the target voice is passive.

The beam search ended if the maximum sentence length (20) was achieved, or if all candidates contained a sentence-final token. This could be a dedicated end-of-sentence token, or a sentence-final punctuation marker (.!?). Instead of directly using the most likely final candidate, we made EATs from all $k$ final candidates and compared them with the original EAT. Specifically, we applied reward or punishment based on the first EAT sequence, which always corresponds to the main verb and its arguments (as explained in Section 3). Here, the first 9 features represent clausal grammatical features (Table 1). To ensure their proper retainment, we punished their deviation from those in the original EAT with 10. In other features we applied a reward or punishment of 1 for identity or divergence. The rationale was to strictly enforce the grammatical class of the transformation, but be more lenient in allowing lexical paraphrases. The candidate with the highest probability after the EAT-comparison was chosen as the final transformation.

## 5.2 Experimental results

In this section we describe the results we received in sentence reproduction (5.2.1) and grammatical transformation across 14 category pairs (5.2.2).

### 5.2.1 Sentence reproduction

We reproduced 1000 random sentences with the trained EAT2seq model. The sentences were taken from a test set not used for training the model, but derived from the same corpora (with the same class distribution). Reproduction constitutes the same task as grammatical transformation, except that no grammatical parameters are altered. The model should repeat the original sentence as faithfully as possible. This is the most basic task demonstrating whether EAT2vec maintains enough information about the original sentence to allow faithful reconstruction at the decoding stage.

We used the BLEU score to evaluate model performance. BLEU is a common metric used for evaluating MT, and is based on n-gram overlap between the candidate translation and a baseline typically produced by a human (Papineni et al., 2002b). It has been shown to correlate with human judgement, especially on larger corpora (Papineni et al., 2002a; Coughlin, 2003). We applied it in measuring reproduction success by treating the original sentence as the target.

As pre-processing for BLEU measurement, we lowercased all sentences, removed punctuation, and made the following replacements to both the target and the reproduction:

n't → not

'm → am

're → are

've → have

'd → would

is → 's (if not in the beginning of the sentence)[19]

The purpose of pre-processing was to avoid unnecessary punishment for abbreviations, and to discard punctuation errors due to their minor effect on readability.

On our 1000 example sentences we received a reproduction BLEU score of 74.45, which can be considered strong. For comparison, in the experiments conducted by Coughlin (2003), a BLEU

---

[19]We mapped *is* to *'s* rather than vice versa due to its ambiguity with the possessive suffix.

score over 60 was systematically correlated with a human translation evaluation of 3 or higher, on a scale from 1 (worst) to 4 (best).

### 5.2.2 Grammatical transformations

In addition to reproducing sentences, we gathered sentences representing different grammatical classes and applied various grammatical transformations to them. We experimented with the following classes: *Force* (declarative, question), *Truth* (affirmed, negated), *Voice* (active, passive), *Tense* (present, past, perfect, pluperfect), and *Aspect* (perfective, imperfective). The classes were detected from the Boolean grammatical features in the sentences' EAT-representations. From our test set, we randomly drew 300 example sentences for each of the 12 grammatical classes. We used two approaches for measuring transformation success: *manual evaluation* and *back-transformation*. Some representative example transformations are provided in Table 2

### 5.2.3 Manual evaluation

In each of the 14 directions, we randomly chose 50 transformations to evaluate manually. We classified each transformed sentence to one of the following categories:

1: No errors

2: Grammatical errors: mistakes in inflection, word order, or grammatical elements

3: Lexical errors: content words missing or incorrect

We further distinguished between errors in the targeted class, and those errors that occur outside of the relevant transfomation. For instance, a missing article in a noun phrase constitutes a grammatical error, but is irrelevant for the success of e.g. tense transformation.

The majority of errors were not fatal to retaining most of the intended meaning. However, we made no assessment of error severity. Even small mistakes like missing articles, agreement errors, etc. were classified to one of the error categories. We made two exceptions to this principle: (i) we discarded punctuation errors, as these can easily be treated in post-processing, and (ii) we discarded interjections and words like *but* or *so* at the beginning of the sentence. Hence, if the original sentence was *But hey, this was good!*, its present tense transformation *This is good.* would have been considered perfect.

Results from the manual evaluation are presented in Table 3. The model succeeded perfectly in 66% of the transformations altogether. Perfect transformations always formed the largest class, and constituted the majority in all cases except two (voice transformation to both directions). The most common error type was a lexical error unrelated to the transformation target. Among both grammatical and lexical errors, the majority occured outside the transformation.

### 5.2.4 Back-transformation

In addition to manual evaluation, we conducted a fully automatic measurement of transformation success, based on the idea of using *back-translation* to evaluate MT (e.g. Rapp 2009). Since we are evaluating the success of grammatical transformations, we call our method *back-transformation*.

As an initial measure of transformation success, we first examined if the transformed sentence belongs to the intended category, based on the Boolean grammatical features in its EAT-representation. If it did not, we considered the transformation as a failure. If it did, we transformed it back to the original category, and measured the BLEU score between this back-transformation and the original sentence. We did not conduct the back-transformation if the first transformation had failed, as this would not measure the transformation under examination. We applied each transformation type to both directions, yielding 14 experiments altogether. However, it bears emphasis that each test measured the combined effect of *two* transformations: one to each direction. Tense was the only class with more than two variants, and here we transformed between the present tense and all others.

| Direction | Original | Transformed |
|---|---|---|
| Declarative-question | Right, but we're not family. | Aren't we family? |
| | Sami gave away millions of dollars. | Did Sami give away lots of dollars? |
| Question-declarative | Did you just burn all of your clothes? | You just burned all of your clothes. |
| | Freaky, isn't it? | It's not freaky. |
| Affirmed-negated | Go back a bit. | Don't go back a bit. |
| | Maria and I broke up. | Maria and I didn't break up. |
| Negated-affirmed | He doesn't even know Ruth Sanders | He even knows Ruth Sanders. |
| | Not for us, it isn't. | It's for us. |
| Active-passive | Look, he missed most of my birthdays anyway. | Most of my birthdays were missed by him, anyway. |
| | Have you altered it? | Has it been altered by you? |
| Passive-active | Luckily I am protected by the mirror | Luckily, the mirror protects me. |
| | The cartographic navigation section has been developed by Planetek Italia srl, Bari. | Bari's srl has developed the navigation cartographic section. |
| Present-past | No, Tutsi, that's not for you! | That wasn't for you, Tutsi. |
| | I do not have courage. | I didn't have courage. |
| Past-present | I turned in my coin changer at the toll booth. | I turn off my coin changer at the toll booth. |
| | You didn't see it from her point of view then. | Then you don't see it from her point of view. |
| Present-perfect | Is he really such a tough guy? | Has he really been such a tough guy? |
| | I am not sleeping in the woods. | I haven't been sleeping in the woods. |
| Perfect-present | Or has a snake bitten you? | Does a snake bite you? |
| | A very important envelope has just been found. | A very important envelope is just found. |
| Present-pluperfect | We don't even go to the supermarket. | We hadn't even gone to the supermarket. |
| | HE'S NOT HURTING ANYONE. | He hadn't been hurting anyone. |
| Pluperfect-present | They had been overcome by smoke. | They're overcome by smoke. |
| | So it had begun. | So it begins. |
| Perfective-imperfective | No, I'm not a girl. | I'm not being a girl. |
| | He collapsed on his back. | He was collapsing on his back. |
| Imperfective-perfective | Are you wearing a bronzer? | Do you wear bronzer? |
| | You're not moving into a palace, for God's sake. | You don't move into a palace, for God's. |

Table 2: Example transformations

Results are presented in Table 4. In addition to the back-transformation BLEU, it contains the success rate of the first transformation (i.e. whether the transformed sentence's class corresponds to the intended class), and the number of back-transformations that are identical with the original sentence (after pre-processing as described in 5.2.1).

| Transformation | Perfect | Grammatical errors | | Lexical errors | |
|---|---|---|---|---|---|
| | | Target class | Elsewhere | Target class | Elsewhere |
| Declarative–question | 35 | 3 | 7 | 1 | 10 |
| Question–declarative | 37 | 4 | 5 | 0 | 7 |
| Affirmed–negated | 27 | 4 | 9 | 0 | 7 |
| Negated–affirmed | 43 | 2 | 5 | 1 | 4 |
| Active–passive | 24 | 10 | 12 | 8 | 16 |
| Passive–active | 22 | 11 | 12 | 7 | 22 |
| Present–past | 38 | 0 | 4 | 0 | 9 |
| Past–present | 34 | 1 | 8 | 0 | 10 |
| Present–perfect | 34 | 2 | 7 | 2 | 11 |
| Perfect–present | 37 | 0 | 5 | 1 | 12 |
| Present–pluperfect | 35 | 2 | 7 | 1 | 8 |
| Pluperfect–present | 27 | 5 | 7 | 2 | 16 |
| Perfective–imperfective | 32 | 7 | 6 | 2 | 11 |
| Imperfective–perfective | 38 | 2 | 7 | 2 | 9 |

Table 3: Manual evaluation of transformations (50 sentences per pair; error overlap possible)

Our EAT2seq implementation was able to correctly transform the grammatical type in practically all cases. From the 4200 transformations altogether, only four failed to be classified to the intended category. Even across two transformations back to the original category, we achieve BLEU scores of 65 or higher in all cases except voice transformation. In 7/14 directions, more than half of the back-transformations are identical to the original sentence. Averaging over all experiments, the identical back-transformation rate is 48%.

The most challenging transformation type was voice, where BLEU scores remained below 60 in both directions. Inferior performance here was likely caused by a lack of exposure to relevant examples during training, as passive sentences containing an Agent made up only around 1% of the training set. Nevertheless, even in the worst case (passive–active), more than 1/4 of the back-transformations were identical to the original.

Overall, most transformations were successful, and indicate that EAT2seq can be used to produce transformations that target in principle any grammatical class, and reliably retain of the original sentence's content. However, the model fared systematically worse in voice transformation than in other cases. Further training would be needed to improve our model's performance on passive sentences containing an Agent argument. We note that this difficulty concerns data imbalance rather than the EAT2seq method itself, and can be remedied by appropriate training.

# 6 Related work

EAT2seq is a "hybrid" approach to NLP, combining both symbolic methods (inspired by formal semantic theory) and data-based methods, with the goal of utilizing the benefits of both frameworks without succumbing to the weaknesses of either. Such combinations have been used in certain approaches to *natural language understanding* (Garrette et al., 2011; Lewis and Steedman, 2013; Beltagy et al., 2016). However, EAT2vec is the first method we know of that uses (approximative) logical form representations as encoder inputs to produce sentence embeddings. Additionally, prior hybrid approaches to semantic encoding are not as such usable for sentence reproduction or transformation. Since EAT2vec is a part of EAT2seq and trained along with it, we are training both sentence embeddings and their linguistic reconstructions at the same time.

The encoder-decoder structure we use instantiates a standard neural machine translation (NMT) architecture, with the exception of replacing word embeddings with EAT-sequences as encoder inputs. Since its introduction, NMT has become the dominant approach to MT (e.g. Luong et al. 2015; Wu et al. 2016). The basic idea of EAT2seq is to refine the NMT input by

| Transformation | Correct target category | Back-transformation | |
|---|---|---|---|
| | | Identical | BLEU |
| Declarative-question | 299/300 | 136/299 | 64.52 |
| Declarative-question | 299/300 | 136/299 | 81.82 |
| Affirmed-negated | 300/300 | 140/300 | 67.95 |
| Negated-affirmed | 300/300 | 164/300 | 79.38 |
| Active-passive | 300/300 | 102/300 | 55.29 |
| Passive-active | 299/300 | 83/299 | 55.55 |
| Present-past | 300/300 | 164/300 | 75.19 |
| Past-present | 300/300 | 148/300 | 70.47 |
| Present-perfect | 300/300 | 157/300 | 72.92 |
| Perfect-present | 299/300 | 159/299 | 76.92 |
| Present-pluperfect | 300/300 | 154/300 | 73.01 |
| Pluperfect-present | 300/300 | 122/300 | 67.28 |
| Perfective-imperfective | 300/300 | 151/300 | 68.28 |
| Imperfective-perfective | 300/300 | 155/300 | 74.51 |

Table 4: Automatic evaluation of transformations
("Identical" = back-transformations identical with the original sentence)

separating between grammatical and thematic information, thus allowing for controlled transformation of either by simple symbol replacement operations. Similar ideas have been presented in prior work on *formality transfer*, which can be considered a type of style transfer (Sennrich et al., 2016; Rao and Tetreault, 2018).

EAT2seq is similar to the *back-translation* approach to style transfer (Prabhumoye et al., 2018), where a style-neutral representation is approximated by an encoding of a machine-translated version of the original sentence. More generally, it continues the line of many recent studies where a latent sentence representation is mapped onto different target styles via either separate decoder RNNs or a single decoder with style features as inputs (Shen et al., 2017; Fu et al., 2018; Shetty et al., 2018). Their main point of difference is that EAT2seq has *two* intermediate representations between the source and target sentence: a symbolic EAT-sequence, and the encoding derived from this sequence. The first of these allows user control, making possible the controlled transformation of the desired output. Further, no additional parallel corpora are required for providing the latent representation, and the model is only trained on monolingual sentences.

In addition to text transformation, stylistic and grammatical features have been used for controlling the output in *text generation* (Hu et al., 2017). Various content- and style-based features were also used by Juuti et al. (2018) for targeted generation of restaurant reviews with an NMT model. These methods bear some similarity to using EAT2seq for text generation (Section 4.4), and adopting an EAT2seq-based approach to similar tasks constitutes a possible line of future work.

Grammatical transformation could of course be undertaken by standard supervised MT, where the source sentence belongs to one grammatical category and the target to another. This would require large separate training sets for each task, and the resulting model would only be applicable for the particular transformation type(s) it is trained on. We thus find it relatively unsurprising that we have not discovered supervised MT systems tailored for very specific grammatical transformations of the type reviewed in Section 5. To our knowledge, EAT2seq is the only method proposed so far that allows *any* grammatical or lexical features (encoded in the EAT-representation) to be transformed, in *any combination*, with a *single model without any task-specific training*.

MT has been used to translate between Abstract Meaning Representations (AMRs) (Banarescu et al., 2013) and natural language text, relying on manually constructed parallel corpora (Ferreira et al., 2017; Gildea et al., 2018; Cohn et al., 2018). On the other hand, LF-representations have been produced directly from the dependency parse Reddy et al. (2016,

2017). EAT2seq partly combines the main insights of these lines of work by first constructing a dependency-based EAT-representation and then training an NMT network to reconstruct an English sentence from it. By automatically creating the parallel corpus, it allows using unlabeled monolingual corpora as training data.

Traditional rule-based NLP methods require no training data, which allows them to be applicable in tasks like detailed grammatical transformations. Rule-based solutions have been proposed for e.g. sentence negation (Ahmed and Lin, 2014; Bilu et al., 2015), style transfer (Khosmood and Levinson, 2008, 2009, 2010; Khosmood, 2012), and producing automatic exercises for language pedagogy (Baptista et al., 2016). EAT2seq also uses rules for producing transformations, but the rules are maximally simple, changing a single Boolean feature per grammatical property. EAT2seq can also be used to change *any combination* of features with only one transformation (e.g. transforming present declaratives into past questions). Finally, by using an NMT network for producing the output, EAT2seq is able to surpass many of the traditional problems in rule-based NLP methods.

# 7 Conclusions and current work

In this paper, we have presented EAT2vec and EAT2seq as methods of sentences encoding and (re)production. We have discussed its applicability to a variety of NLP tasks, and provided empirical evidence for its power in producing grammatical transformations. EAT2seq has significantly less training data requirements than alternative approaches, and the *same* model can be used for all tasks without separate training. Our experimental results indicate that the framework is promising and can be implemented using existing tools (such as Spacy).

The EAT2seq framework is unique in combining formal semantic methods in data preprocessing, and NMT for transforming abstract semantic representation into English. Effectively, it allows the user to go from English to EAT-representations and back, applying a variety of transformations in both grammatical and lexical aspects. The sentences it produces are realistic, and go beyond simple text-book examples of transforming logical forms into English. This makes EAT2seq unique among logic-inspired systems in being tailored to real-word text.

We believe that EAT2seq has potential to be applied especially in tasks where training data is scarce. Further, by allowing a great variety of linguistic transformations without requiring any labeled data, separate training, or transformation-specific rules, EAT2seq markedly increases the practical feasibility of many transformation tasks that have been difficult to perform so far. It thus widens the range of feasible NLP tasks without furthering training data requirements.

As EAT2seq is a general approach that can be used for a number of tasks, future work can take many directions. We are currently working on an EAT2seq-based approach to style transfer, focusing on methods that allow any target style to be mimicked with only one model. Another promising venue is the application of EAT2seq to MT, in both supervised and unsupervised settings. Related to MT, it is important to extend EAT2seq beyond English. We are also conducting more systematic and exhaustive evaluations of EAT2seq's effectiveness overall.

# 8 Acknowledgements

# A Detailed mapping to Neo-Davidsonian logical forms

In this appendix we provide a more rigorous theoretical account of how our EAT-vectors relate to logical form representations in Neo-Davidsonian semantic theory. Understanding this is not

necessary for practical considerations, but provides an explicit demonstration of how the EAT-vectors are interpreted.

A major benefit of the Neo-Davidsonian scheme is that all EATs are *conjunctions* of simple predications or relations. This means that a sequence is always interpreted as the conjunction of the interpretations of its elements. However, as a conjunction entails all of its conjuncts individually, whenever an inference from a complex expression to some of its parts is invalid, the conjunctive scheme is falsified. Therefore, to retain a conjunctive interpretation of an EAT-sequence, we must map each EAT-vector to an EAT-interpretation in a way that avoids such entailment failures. We deal with such entailment problems concerning modification structures in Section A.1. Our strategy here is to liberalize the EAT-vector interpretation to allow words not to predicate directly.

Another issue concerns the implicit existential quantification of events and their arguments. Evidently, this is not the case in conditionals, modal propositions, or contents of propositional attitudes. Further, it is unclear how the strictly conjunctive scheme can deal with conditionals, or connectives in general. Section A.2 modifies the interpretation to allow such cases without breaking away from the conjunctive interpretation of EAT-sequences.

## A.1   Non-entailing modifiers

We began with a simple mapping from word triplets to Event-Agent-Theme relations, and then added markers of truth/negation, tense, illocutionary force, and various grammatical markers. The thematic mapping is reproduced below.

(30)   $<W_1, W_2, W_3> \Leftrightarrow W_1(e) \wedge W_2(x) \wedge W_3(y) \wedge \text{Ag}(e, x) \wedge \text{Th}(e, y)$

We take the empty token $\varnothing$ simply to lack an interpretation.

(31)   $<W_1, W_2, \varnothing> \Leftrightarrow W_1(e) \wedge W_2(x) \wedge \text{Ag}(e, x)$

Agent and Theme arguments retain their thematic status even if Event is empty, which can only be the case if the Event is repeated, i.e. with modification of Agent or Theme. Hence, an empty Event still implies the existence of an event, while an empty Agent or Theme does not imply the respective argument's tacit existence.

(32)   A brown dog saw a black cat
       <see, dog, cat> <$\varnothing$, brown, black>

Even though the simple mapping from EAT-vectors to Neo-Davidsonian LFs can handle basic adjectival or adverbial modification, problems arise with compounds. Treating compound elements as conjoined gives evidently false truth-conditions, as shown by the invalidity of inferences like *ant eater* $\Rightarrow$ *ant*. However, as mentioned in Section 3.1.2, we represented them as such to avoid complications. It turns out, however, that we can allow the simple vector representation of compounds as modifiers in a more rigorous manner, if we alter the basic mapping between EAT-vectors and Neo-Davidsonian LFs from (30) to (33).

(33)   $<W_1, W_2, W_3> \Leftrightarrow$
       $W_1(e) \wedge W_2(x) \wedge W_3(y) \wedge R_1(e, e') \wedge R_2(x, x') \wedge R_3(y, y') \wedge \text{Ag}(e', x') \wedge \text{Th}(e', y')$
       where $R_1$, $R_2$ and $R_3$ are some relations relevant in the discourse/context.

That is, a triplet $<W_1, W_2, W_3>$ expresses that the Event bears some contextually relevant relation to some event that has the property $W_1$, and the same for the Agent and Theme *mutatis mutandis*. In the basic case assimilating to (30), $\{R_1, R_2, R_3\}$ all are the *identity* relation. However, with other modifiers, such as those appearing in compounds, the relation can be something else, such as 'tends-to-eat' for *ant* in *ant eater*. The EAT-vector does not specify the nature of this relation, since the grammar does not do this either: the semantic association between *ant* and *ant eater* is completely different from that between *police* and *police dog*, for instance. The only property that unites these relations is that they are contextually relevant for determining the nature

of the entity under discussion. We have now removed the problem mentioned above, as inferences like *ant eater* $\Rightarrow$ *bears a contextually relevant relation to (some) ant(s)* are valid.[20]

Notationally, we can abbreviate (33) by changing the lexical predicates $\{W_1, W_2, W_3\}$ into $\{W*_1, W*_2, W*_3\}$ according to the scheme (34).

(34)    $W*(x) \Leftrightarrow \exists y[W(y) \wedge R(x,y)]$
        where $W$ is a standard lexical predicate, and $R$ is some contextually relevant relation

For instance, the word *dog* no longer stands for the basic predicate $\mathrm{DOG}(x)$ denoting a type of animal, but the more complex predicate $\mathrm{DOG}*(x) \Leftrightarrow \exists y[\mathrm{DOG}(y) \wedge R(x,y)]$, where $R$ is some contextually relevant relation. Treating all lexical predicates along the lines of (34) allows us to retain the simpler notation from the original (30), with the difference that the "$*$"-version of each lexical predicate is used.

(35)    $<W_1, W_2, W_3> \Leftrightarrow W*_1(e) \wedge W*_2(x) \wedge W*_3(y) \wedge \mathrm{Ag}(e, x) \wedge \mathrm{Th}(e, y)$
        where $\forall i : W*_i(x) \Leftrightarrow \exists y[W_i(y) \wedge R(x,y)]$, and $R$ is some contextually relevant relation

The "$*$" can be thought of as always being implicitly present in the EAT-examples given in this paper.

## A.2    Existential quantification

So far we have assumed that all elements in an EAT-sequence represent predicates of existentially quantified variables.

(36)    A brown dog saw a cat
        $<$see, dog, cat$>$
        $\exists e \exists x \exists y \, [\mathrm{SEE}(e) \wedge \mathrm{DOG}(x) \wedge \mathrm{CAT}(y) \wedge \mathrm{Ag}(e, x) \wedge \mathrm{Th}(e, y)]$

Problems for this simple analysis are created by modal adverbs, conditionals, and propositional attitude ascriptions, none of which entail the existence of the event. Taking inspiration from Discourse Representation Theory (Kamp, 1981, 1995), in this section we modify the EAT-vector interpretation to allow such cases within a rigorously conjunctive formalism.

We have treated modal auxiliaries as simple event predicates: *can jump* $\Leftrightarrow$ $\mathrm{CAN}(e) \wedge$ $\mathrm{JUMP}(e)$. As inferences like *can jump* $\Rightarrow \exists e \, \mathrm{JUMP}(e)$ are invalid, this cannot be correct, if $e$ is an existentially quantified variable ranging over events.

Another problem is manifested by conditionals, such as *if*-clauses: *Mary runs if John walks* does not entail *Mary runs*. Classical Fregean logic has no trouble with such constructions, as they exhibit the relation of ("material") *implication*: the truth of the conditioned proposition is dependent on the truth of the conditional. (Neo-)Davidsonians can also take the implication relation to hold between existential quantifications of events.

(37)    Mary runs if John walks
        $\exists e \, [\mathrm{WALK}(e) \wedge \mathrm{Ag}(e, \mathrm{J})] \rightarrow \exists e \, [\mathrm{RUN}(e) \wedge \mathrm{Ag}(e, \mathrm{M})]$

However, the strict conjunctive scheme of our EAT-sequence interpretation prevents such implication structures, as everything must be expressed as a conjunction of simple predications. One candidate for a conjunctive reconstruction of (36) might be to think of implication as a dyadic relation holding between two event variables.

---

[20]Of course, this comes at the cost of liberalizing the interpretation, as $\{R_1, R_2, R_3\}$ can in principle range over any conceivable relations. However, such indeterminacy of lexical meaning is well-attested property of language that is manifested not only in compounds but all words. "Non-literal" uses of words like metaphor, exaggeration, and so-called *ad hoc concepts* demonstrate that words have a malleable semantic interpretation that allows dissociation from their standard meaning in certain contextually appropriate (but nondeterministic) ways (e.g. Carston 2002). Hence, a looser mapping between words and concepts is required anyway, independently of considerations of non-entailing modifiers.

(38)  Mary runs if John walks
      $\exists e \exists e'$ [WALK$(e) \wedge$ Ag$(e,$ J$) \wedge$ RUN$(e') \wedge$ Ag$(e',$ M$) \wedge$ IF$(e, e')$]

However, it is easy to see that (37) fails to be equivalent to (38), as the latter entails that Mary runs while the former does not. Nevertheless, we continue to maintain the analysis of connectives as dyadic relations, and instead modify the existence assumption.

Discourse Representation Theory (DRT) (Kamp, 1981, 1995) maintains that semantic interpretation consists in building *Discourse Representation Structures* (DRSs), to which elements are dynamically added when the discourse unfolds. A DRS consists of a list of *Discourse Referents* (DRs), and a list of *DRS conditions*, which are predications over DRs. An example DRS with traditional Fregean verb semantics is provided in (39), using the standard box notation of DRT.

(39)  A dog runs

      DRS | DR: $x$ | Cond: DOG$(x)$, RUN$(x)$

We can further modify the DRSs to use Neo-Davidsonian semantics by adding separate event DRs. We also remind that the "*"-interpretation defined in (34) is implicitly assigned for every lexical predicate, as discussed in Section A.1.

(40)  DRS | DR: $e, x$ | Cond: RUN$(e)$, DOG$(x)$, Ag$(e, x)$

A DRT-reformulation of the thematic mapping (30) is provided in (41).

(41)  $<W_1, W_2, W_3> \Leftrightarrow$

      DRS | DR: $e, x, y$ | Cond: $W_1(e)$, $W_2(x)$, $W_3(y)$, Ag$(e, x)$, Th$(e, x)$

Additionally, a DRS can relate to another DRS via relations corresponding to propositional connectives in classical logic. In DRT, these are called *complex DRS-conditions* (Kamp, 1981, 1995). DRSs can be *sub-DRSs* of larger DRSs that express the propositional relations. A representation for a basic conditional clause is provided in (42), where DRS$_1$ and DRS$_2$ are sub-DRSs of DRS$_3$.

(42)  A dog runs, if a cat walks

      DRS$_1$ | DRs: $e, x$ | Cond: RUN$(e)$, DOG$(x)$, Ag$(e, x)$

      DRS$_2$ | DRs: $e', y$ | Cond: WALK$(e')$, CAT$(y)$, Ag$(e', x)$

      DRS$_3$ | Cond: IF(DRS$_1$, DRS$_2$)

Let **D** be the DRS of the whole sentence, including all sub-DRSs. We can now think of each EAT-vector as specifying properties of *some* DRS contained in **D**. The original problem of existential quantification no longer arises, since existential commitment is not made for the DRs of sub-DRSs. In the EAT-vector, we mark clausal connectives with the same Boolean parameter as Event-prepositions. We can treat the difference between these as lexical: Event-prepositions link the Event to some other entity, whereas connectives link the whole DRS to another DRS. This is a minor concession to the "lexicalist" view that allocates combinatorially relevant semantic properties to word-internal features. However, since ambiguity never arises between these classes (as clausal connectives and prepositions do not overlap), this theoretical detail has no practical effect on the "non-lexicalist" foundation of EAT2seq.

Certain non-entailing modifiers can also be analyzed as complex DRS-conditions, such as negation, modal auxiliaries (*can, could, might*), or modal adverbs (*possibly, maybe, unlikely*). Intuitively, these specify properties of propositional structures like DRSs, and hence are straightforwardly analyzable as monadic complex DRS-conditions. We can thus justify their treatment on par with other modifiers, provided that the status of a particular modifier as a simple or complex DRS-condition remains a matter of its lexical/conceptual specification.

# References

Afroza Ahmed and King Ip Lin. 2014. Negative Sentence Generation by Using Semantic Heuristics. In *The 52nd Annual ACM Southeast Conference (ACMSE 2014)*.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1881–1890.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *CoRR*, abs/1710.11041.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING/ACL*, pages 86–90.

Mark Baker. 2003. *Lexical Categories*. Cambridge University Press, Cambridge.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob Kevin Knight, Philipp, Koehn, Martha, Palmer, and Nathan Schneider. 2013. Abstract meaning representationfor sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Jorge Baptista, Sandra Lourenco, and Nuno Mamede. 2016. Automatic generation of exercises on passive transformation in Portuguese. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 4965–4972.

Valerio Basile. 2015. *From Logic to Language : Natural Language Generation from Logical Forms*. Ph.D. thesis.

I. Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *The special issue of Computational Linguistics on Formal Distributional Semantics*, 42(4):763–808.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*, pages 1533–1544.

Yonatan Bilu, Daniel Hershcovich, and Noam Slonim. 2015. Automatic claim negation: why, how and when. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 84–93.

George Boolos. 1984. To be is to be the value of a variable (or the values of some variables). *Journal of Philosophy*, 81:430–450.

Hagit Borer. 2005. *The Nominal Course of Events: Structuring Sense, Volume II*. Oxford University Press, Oxford.

Johan Bos. 2008. Wide-Coverage Semantic Analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings, volume 1 of Research in Computational Semantics*, pages 277–286. College Publications.

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

Michael Brennan, Sadia Afroz, and Rachel Greenstadt. 2012. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security*, 15(3):1–22.

Michael Brennan and Rachel Greenstadt. 2009. Practical Attacks Against Authorship Recognition Techniques. In *Proceedings of the Twenty-First Conference on Innovative Applications of Artificial Intelligence*.

Robyn Carston. 2002. *Thoughts and Utterances: The Pragmatics of Explicit Communication*. Blackwell, Oxford.

H. Castañeda. 1967. Comments. In N. Resher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press, Pittsburgh.

Antonio Castro and Brian Lindauer. 2013. Author Identification on Twitter. In *Third IEEE International Conference on Data Mining*, pages 705–708.

Alonzo Church. 1936. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:354–363.

Trevor Cohn, Gholamreza Haffari, and Daniel Beck. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the Association for Computational Linguistics*, pages 273–283.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332.

Deborah A. Coughlin. 2003. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT Summit IX*, page 2327.

Donald Davidson. 1967. The logical form of action sentences. In N. Resher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press, Pittsburgh.

Bonnie Dorr. 1993. *Machine Translation: A View from the Lexicon*. MIT Press, Cambridge.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of The 10th International Natural Language Generation conference*, pages 1–10.

Gottlob Frege. 1879. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Louis Nebert, Halle a. S.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of AAAI*.

D. Garrette, K. Erk, and R. Mooney. 2011. Integrating logical representations with probabilistic information using markov logic. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, pages 105–114.

Daniel Gildea, Zhiguo Wang, Yue Zhang, and Linfeng Song. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the Association for Computational Linguistics*, pages 1616–1626.

Kenneth Hale and Samuel Jay Keyser. 2002. *Prolegomenon to a Theory of Argument Structure*. MIT Press, Cambridge.

Hans van Halteren, R. Harald Baayen, Fiona Tweedie, Marco Haverkort, and Anneke Neijt. 2005. New machine learning methods demonstrate the existence of a human stylome. *Journal of Quantitative Linguistics*, 12(1):65–77.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell, Cornwall.

James Higginbotham. 1985. On semantics. *Linguistic Inquiry*, 16:547–593.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Norbert Hornstein and Paul Pietroski. 2009. Basic operations: Minimal syntax-semantics. *Catalan Journal of Linguistics*, 8:113–139.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

J. Hutchins. 2010. Machine translation: a concise history. *Journal of Translation Studies*, 13(1–2):29–70.

Patrick Juola. 2013. Stylometry and immigration: A case study. *Journal of Law and Policy*, 21(2):287–298.

Mika Juuti, Bo Sun, Tatsuya Mori, and N. Asokan. 2018. Stay on-topic: Generating context-specific fake restaurant reviews. In Javier Lopez, Jianying Zhou, and Miguel Soriano, editors, *Proceedings of the 23rd European Symposium on Research in Computer Security (ESORICS)*, pages 132–151. Springer-Verlag.

Hans Kamp. 1981. A theory of truth and semantic representation. In T.M.V. Janssen J.A.G. Groenendijk and M.B.J. Stokhof, editors, *Formal methods in the Study of Language, Mathematical Centre Tracts 135*, pages 277–322. Mathematisch Centrum, Amsterdam.

Hans Kamp. 1995. Discourse representation theory. In J.-O. stman J. Verschueren and J. Blommaert, editors, *Handbook of Pragmatics*, pages 253–257. John Benjamins, Amsterdam.

Foaad Khosmood. 2012. Comparison of sentence-level paraphrasing approaches for statistical style transformation. In *Proceedings of the 2012 International Conference on Artificial Intelligence*.

Foaad Khosmood and Robert Levinson. 2008. Automatic natural language style classification and transformation. In *Proceedings of the 2008 BCS-IRSG Conference on Corpus Profiling*, page 3.

Foaad Khosmood and Robert Levinson. 2009. Toward automated stylistic transformation of natural language text. In *Proceedings of the Digital Humanities*, pages 177–181.

Foaad Khosmood and Robert Levinson. 2010. Automatic synonym and phrase replacement show promise for style transformation. In *The Ninth International Conference on Machine Learning and Applications*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *CoRR*, abs/1804.07755.

Richard Larson and Gabriel Segal. 1995. *Knowledge of Meaning: An Introduction to Semantic Theory*. MIT Press, Cambridge, MA.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Terje Lohndal. 2014. *Phrase Structure and Argument Structure: A Case Study in the Syntax-Semantics Interface*. Oxford University Press, Oxford.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Association for Computational Linguistics (ACL)*, Berlin, Germany.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223, Reykjavik. ELRA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119, USA. Curran Associates Inc.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Richard Montague. 1970. English as a formal language. In Bruno Visentini, editor, *Linguaggi nella Società e nella Tecnica*, pages 189–224. Edizioni di Communita, Milan.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *Proc. 2012 IEEE Symposium on Security and Privacy*, pages 300–314.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

K. Papineni, S. Roukos, T. Ward, J Henderson, and F. Reeder. 2002a. Corpus-based comprehensive and diagnostic mt evaluation: Initial arabic, chinese, french, and spanish results. In *Proceedings of Human Language Technology*, pages 132–137.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002b. Bleu: a method for automatic evaluation of machine translation. In *ACL-2002: 40th Annual meeting of the Association for Computational Linguistics*, pages 311–318.

Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press, Cambridge.

Barbara H. Partee, Alice T. Meulen, and Robert A. Wall. 1990. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

David Perlmutter. 1978. Impersonal passives and the unaccusative hypothesis. In *Proceedings of the 4th Annual Meeting of the Berkeley Linguistics Society*, pages 157–189, UC Berkeley.

Paul Pietroski. 2003. Quantification and second-order monadicity. *Philosophical Perspectives*, 17:259–298.

Paul Pietroski. 2005. *Events and Semantic Architecture*. Oxford University Press, Oxford.

Paul Pietroski. 2018. *Conjoining Meanings: Semantics without Truth-values*. Oxford University Press, Oxford.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Gillian Ramchand. 2008. *Verb Meaning and the Lexicon: A First Phase Syntax*. Cambridge University Press, Cambridge, MA.

Sudha Rao and Joel R. Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*, pages 129–140.

Reinhard Rapp. 2009. The back-translation score: automatic mtevaluation at the sentence level without reference translations. In *Proceedings of the ACL-IJCNLP Conference Short Papers*, pages 133–136.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. volume 4, pages 127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101. Association for Computational Linguistics.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice (revised ed.)*. International Computer Science Institute, Berkeley, CA.

Bertrand Russell. 1905. On denoting. *Mind, New Series*, 56(14):479–493.

Barry Schein. 1993. *Plurals*. MIT Press, Cambridge.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *NAACL-HLT*, pages 35–40.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A4nt: Author attribute anonymity by adversarial training of neural machine translation. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*, pages 1633–1650, Baltimore, MD. USENIX Association.

Louis Tesnière. 1959. *Èléments de syntaxe structurale*. Klincksieck, Paris.

John Wilkins. 1668. *An Essay Towards a Real Character and a Philosophical Language*. S. Gellibrand.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework of authorship identification for online messages: Writing style features and classification techniques. *Journal American Society for Information Science and Technology*, 57(3):378–393.