Structural biases for semantic prediction

Preliminary work done at LIG

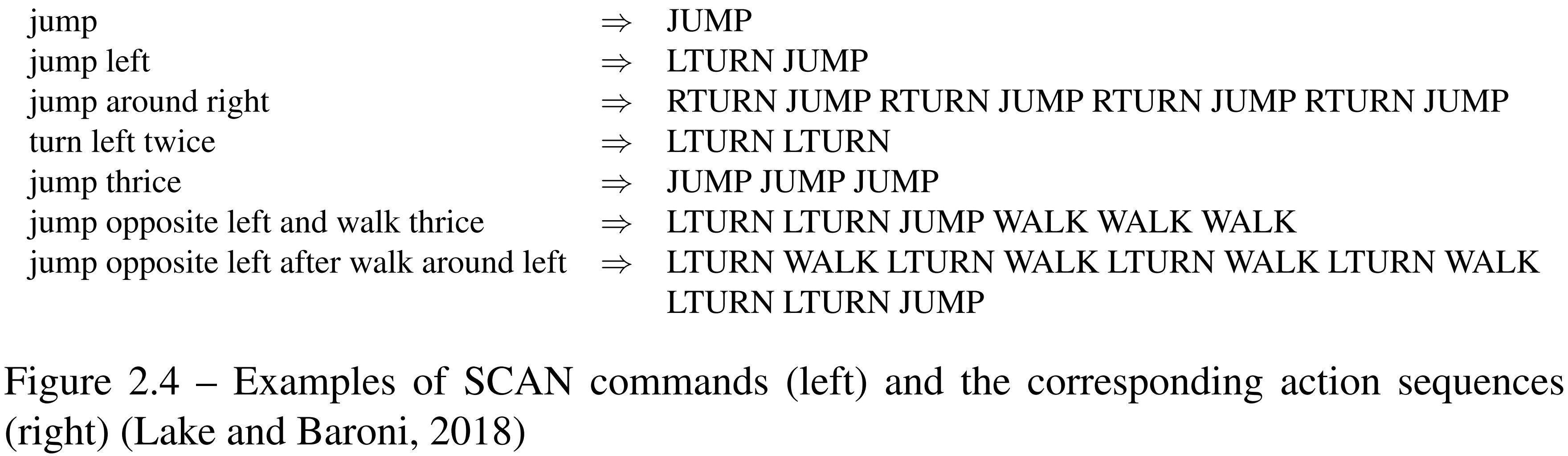
D. Popa, A. Laskina, P. Chem, L. Boualili

L. Besacier, M. Coavoux, E. Devijver, E. Gaussier

Investigating a few ideas on SCAN and COGS

- Reproducing known results
- Integrating syntactic annotation
- Investigating the permutation equivariance framework
  - Extension to large vocabularies
  - Analogy with data augmentation
- Investigating ’’structured’’ spaces (hyperbolic embeddings)

SCAN



jump	⇒	JUMP
jump left	⇒	LTURN JUMP
jump around right	⇒	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	⇒	LTURN LTURN
jump thrice	⇒	JUMP JUMP JUMP
jump opposite left and walk thrice	⇒	LTURN LTURN JUMP WALK WALK WALK
jump opposite left after walk around left	⇒	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

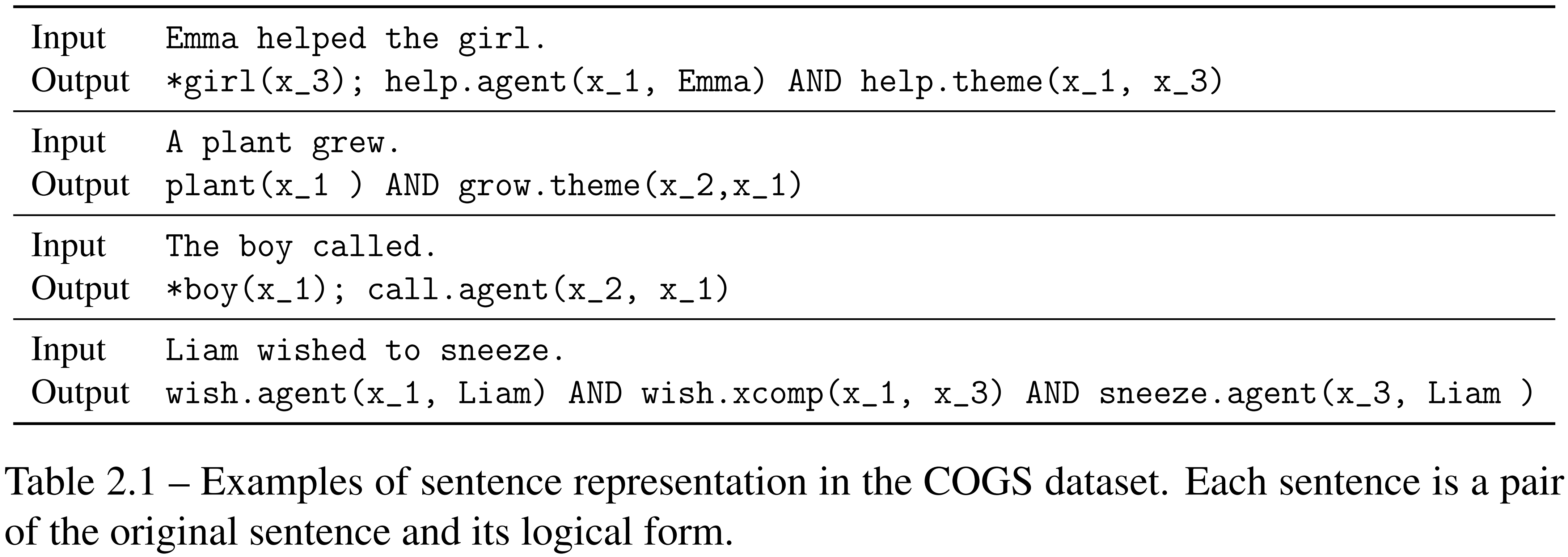
Figure 2.4 – Examples of SCAN commands (left) and the corresponding action sequences (right) (Lake and Baroni, 2018)

The SCAN tasks

In order to test different compositionality capabilities, the SCAN dataset can be split into training and testing in different ways. Specifically the splits include the following:

- *Simple split*: training and testing data are split randomly
- *Add primitive split*: a primitive command (*e.g.*, "turn left" or "jump") is held out of the training set, except in its most basic form (*e.g.*, "jump" $\rightarrow$ JUMP). In the test set, all examples contain this primitive command in a combination with other commands. In other words, the model must learn how to combine a primitive command that it has only seen in isolation.
- *Length split*: the training set includes only commands with action sequence length in the output language shorter than 24 actions, and the test set contains all commands with action sequences longer or equal to 24 actions.

COGS



Input	Emma helped the girl.
Output	*girl(x_3); help.agent(x_1, Emma) AND help.theme(x_1, x_3)
Input	A plant grew.
Output	plant(x_1) AND grow.theme(x_2,x_1)
Input	The boy called.
Output	*boy(x_1); call.agent(x_2, x_1)
Input	Liam wished to sneeze.
Output	wish.agent(x_1, Liam) AND wish.xcomp(x_1, x_3) AND sneeze.agent(x_3, Liam)

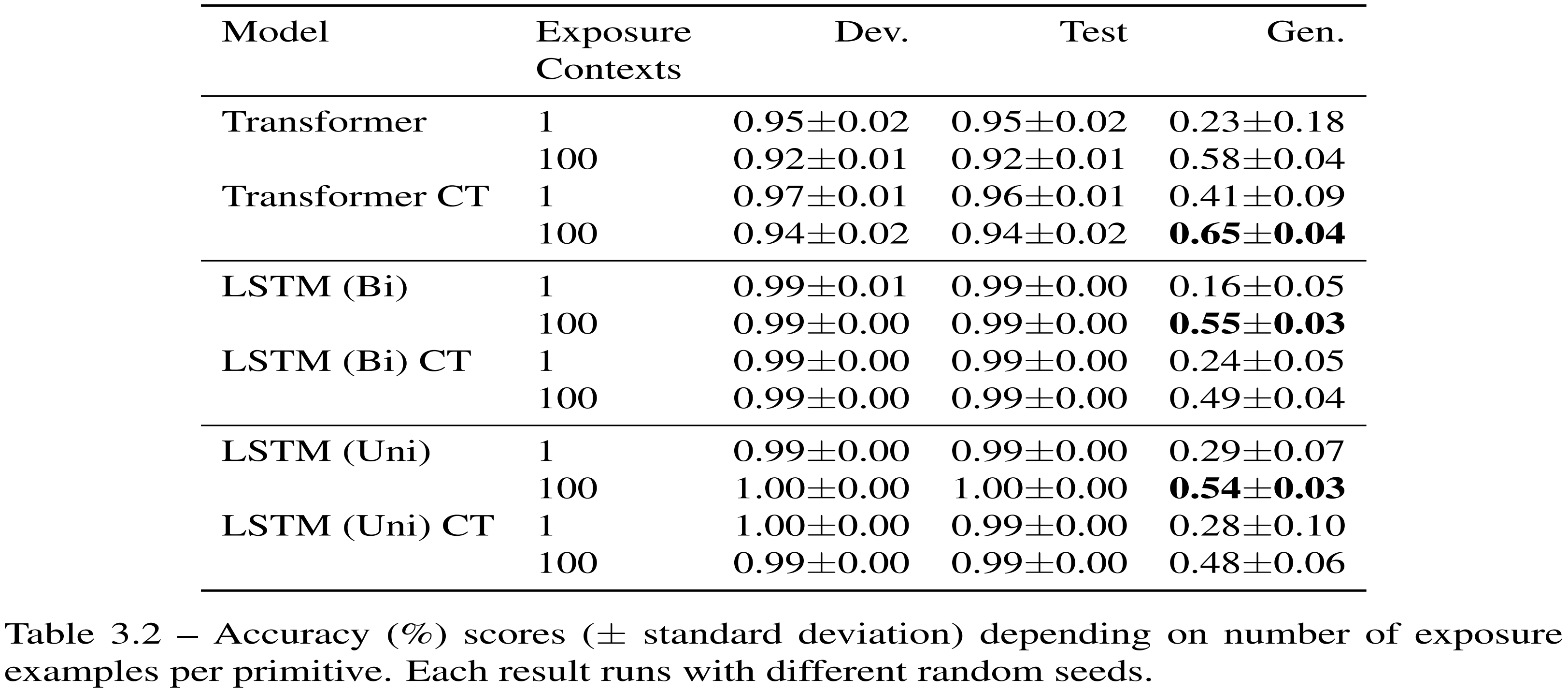
Table 2.1 – Examples of sentence representation in the COGS dataset. Each sentence is a pair of the original sentence and its logical form.

Case	Training	Generalization
Novel Combination of Familiar Primitives and Grammatical Roles		
Subject → Object (common noun)	A **hedgehog** ate the cake.	The baby liked the **hedgehog**.
Subject → Object (proper noun)	**Lina** gave the cake to Olivia.	A hero shortened **Lina**
Object → Subject (common noun)	Henry liked a **cockroach**.	The **cockroach** ate the bat.
Object → Subject (proper noun)	The creature grew **Charlie**.	**Charlie** worshipped the cake.
Primitive noun → Subject (common noun)	**shark**	A **shark** examined the child.
Primitive noun → Subject (proper noun)	**Paula**	**Paula** sketched William.
Primitive noun → Object (common noun)	**shark**	A chief heard the **shark**.
Primitive noun → Object (proper noun)	**Paula**	The child helped **Paula**.
Primitive verb → Infinitival argument	**crawl**	A baby planned to **crawl**.
Novel Combination Modified Phrases and Grammatical Roles		
Object modification → Subject modification	Noah ate **the cake on the plate**.	**The cake on the table** burned.
Deeper Recursion		
Depth generalization: Sentential complements	Emma said **that** Noah knew **that** the cat danced.	Emma said **that** Noah knew **that** Lucas saw **that** the cat danced.
Depth generalization: PP modifiers	Ava saw the ball **in the bottle on the table**.	Ava saw the ball **in the bottle on the table on the floor**.
Novel Verb Argument Structure Alternation		
Active → Passive	The crocodile **blessed** William.	A muffin **was blessed**.
Passive → Active	The book **was squeezed**.	The girl **squeezed** the strawberry.
Object-omitted transitive → Transitive	Emily **baked**.	The giraffe **baked a cake**.
Unaccusative → Transitive	The glass **shattered**.	Liam **shatterd** the jigsaw.
Double object dative → PP dative	The girl **teleported** Liam the cookie.	Benjamin **teleported** the cake to Isabella.
PP dative → Double Object Dative	Jane shipped the cake to John.	Jane shipped John the cake
Verb Class		
Agent NP → Unaccusative subject	The **cobra** helped a dog.	The cobra **froze**.
Theme NP → Object-omitted transitive subject	The hippo **decomposed**.	The hippo **painted**.
Theme NP → Unergative subject	The hippo **decomposed**.	The hippo **giggled**

Table 2.3 – A full list of generalization cases (Kim and Linzen, 2020)

Integrating syntactic annotation

```
( TOP ( S ( NP ( NNP Emma ) ) ( VP ( VBD helped ) ( NP ( DT the ) ( NN
                  girl ) ) ) ( .  .  )  )  ),
```



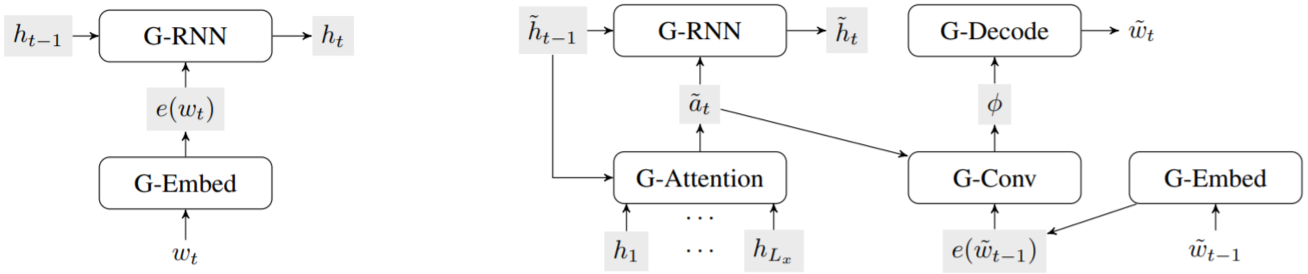
Model	Exposure Contexts	Dev.	Test	Gen.
Transformer	1	0.95±0.02	0.95±0.02	0.23±0.18
	100	0.92±0.01	0.92±0.01	0.58±0.04
Transformer CT	1	0.97±0.01	0.96±0.01	0.41±0.09
	100	0.94±0.02	0.94±0.02	**0.65±0.04**
LSTM (Bi)	1	0.99±0.01	0.99±0.00	0.16±0.05
	100	0.99±0.00	0.99±0.00	**0.55±0.03**
LSTM (Bi) CT	1	0.99±0.00	0.99±0.00	0.24±0.05
	100	0.99±0.00	0.99±0.00	0.49±0.04
LSTM (Uni)	1	0.99±0.00	0.99±0.00	0.29±0.07
	100	1.00±0.00	1.00±0.00	**0.54±0.03**
LSTM (Uni) CT	1	1.00±0.00	0.99±0.00	0.28±0.10
	100	0.99±0.00	0.99±0.00	0.48±0.06

Table 3.2 – Accuracy (%) scores (± standard deviation) depending on number of exposure examples per primitive. Each result runs with different random seeds.

Investigating permutation equivariance

$$\begin{tikzcd}
x \arrow[r, dashed, "T_g"] \arrow[d, "\phi"'] & x' \arrow[d, "\phi"] \\
y \arrow[r, dashed, "T'_g"] & y'
\end{tikzcd}$$

$$\begin{tikzcd}
\textit{WALK LEFT AND RUN} \arrow[r, dashed] \arrow[d] & \textit{JUMP LEFT AND RUN} \arrow[d] \\
\textit{LTURN WALK RUN} \arrow[r, dashed] & \textit{LTURN JUMP RUN}
\end{tikzcd}$$

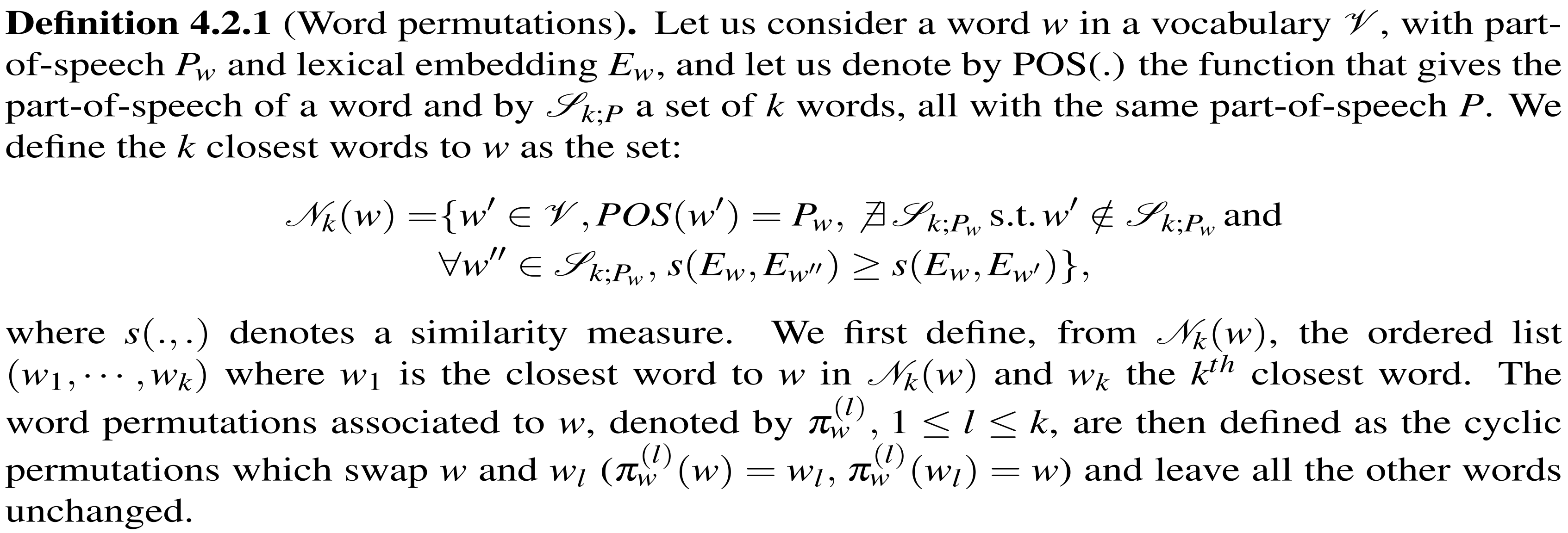


(a) G-Equivariant Encoder

(b) G-Equivariant Decoder

Figure 4.2 – Architecture of equivariant seq2seq model by Gordon et al. (2020)

Extension to large vocabularies (1)

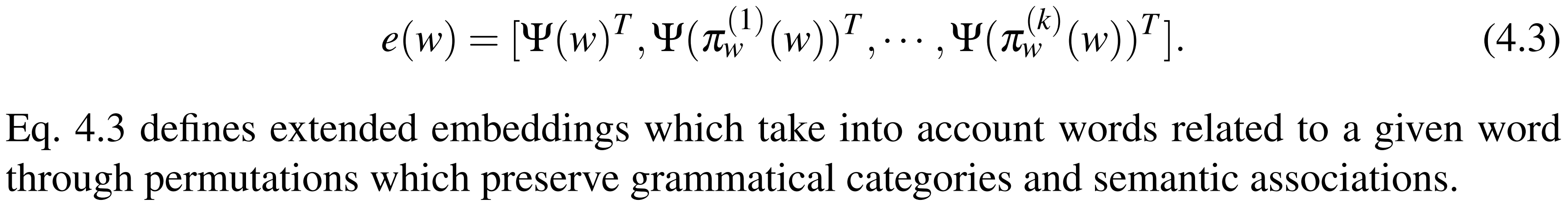


Definition 4.2.1 (Word permutations)**.** Let us consider a word w in a vocabulary $\mathscr{V}$, with part-of-speech P_w and lexical embedding E_w, and let us denote by POS(.) the function that gives the part-of-speech of a word and by $\mathscr{S}_{k;P}$ a set of k words, all with the same part-of-speech P. We define the k closest words to w as the set:

$$\mathscr{N}_k(w) = \{w' \in \mathscr{V}, POS(w') = P_w,\ \nexists \mathscr{S}_{k;P_w} \text{ s.t. } w' \notin \mathscr{S}_{k;P_w} \text{ and } \forall w'' \in \mathscr{S}_{k;P_w},\ s(E_w, E_{w''}) \geq s(E_w, E_{w'})\},$$

where $s(.,.)$ denotes a similarity measure. We first define, from $\mathscr{N}_k(w)$, the ordered list $(w_1, \cdots, w_k)$ where w_1 is the closest word to w in $\mathscr{N}_k(w)$ and w_k the k^{th} closest word. The word permutations associated to w, denoted by $\pi_w^{(l)}$, $1 \leq l \leq k$, are then defined as the cyclic permutations which swap w and w_l ($\pi_w^{(l)}(w) = w_l$, $\pi_w^{(l)}(w_l) = w$) and leave all the other words unchanged.

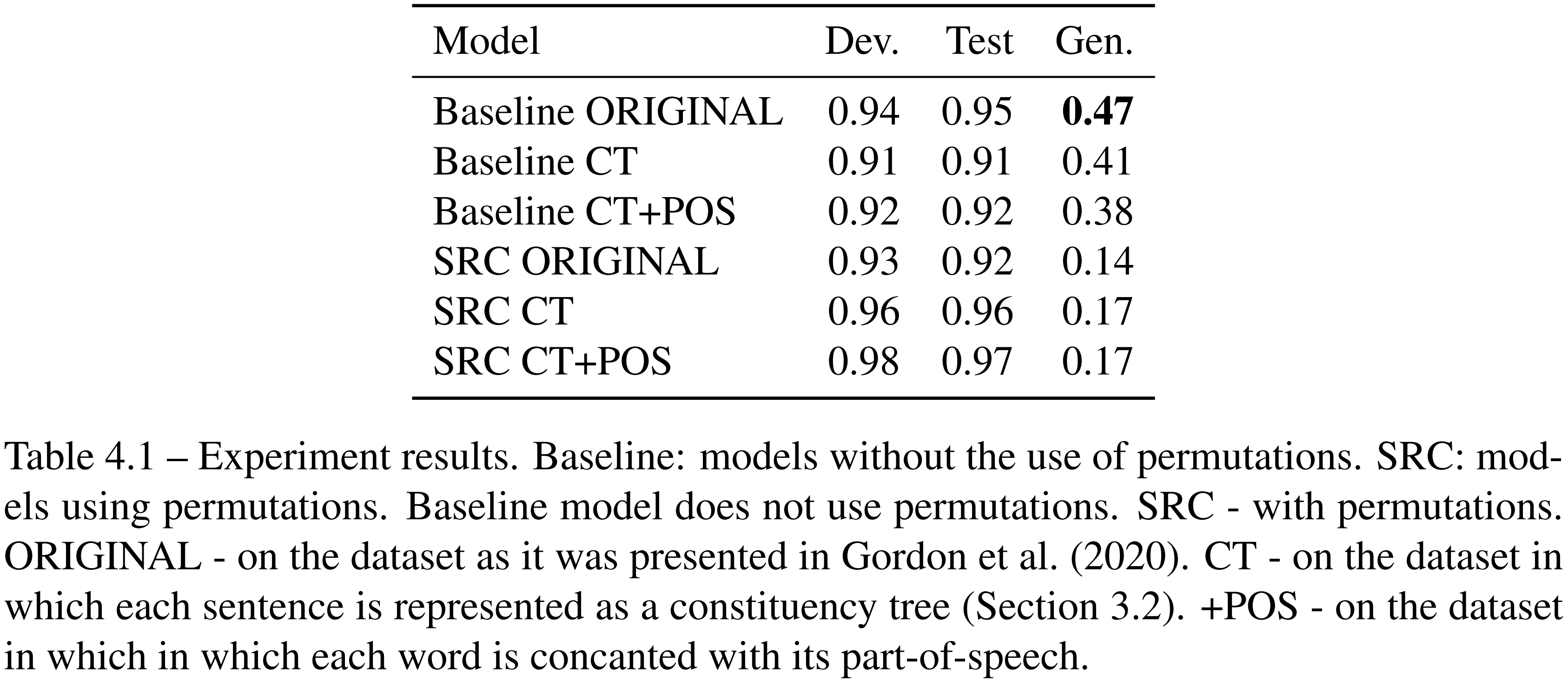
Extension to large vocabularies (2)



$$e(w) = [\Psi(w)^T, \Psi(\pi_w^{(1)}(w))^T, \cdots, \Psi(\pi_w^{(k)}(w))^T]. \tag{4.3}$$

Eq. 4.3 defines extended embeddings which take into account words related to a given word through permutations which preserve grammatical categories and semantic associations.

Extension to large vocabularies (3)

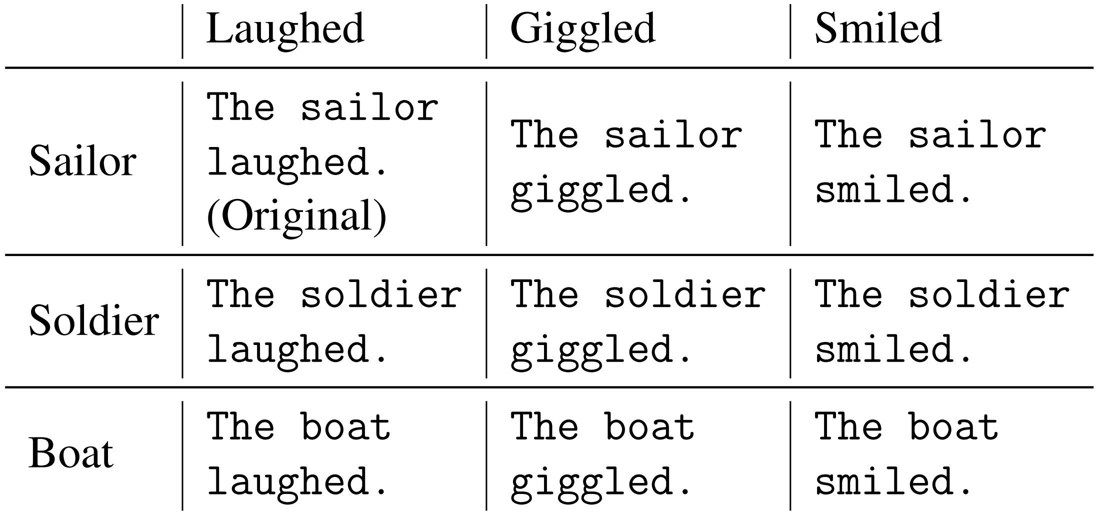


Model	Dev.	Test	Gen.
Baseline ORIGINAL	0.94	0.95	**0.47**
Baseline CT	0.91	0.91	0.41
Baseline CT+POS	0.92	0.92	0.38
SRC ORIGINAL	0.93	0.92	0.14
SRC CT	0.96	0.96	0.17
SRC CT+POS	0.98	0.97	0.17

Table 4.1 – Experiment results. Baseline: models without the use of permutations. SRC: models using permutations. Baseline model does not use permutations. SRC - with permutations. ORIGINAL - on the dataset as it was presented in Gordon et al. (2020). CT - on the dataset in which each sentence is represented as a constituency tree (Section 3.2). +POS - on the dataset in which in which each word is concanted with its part-of-speech.

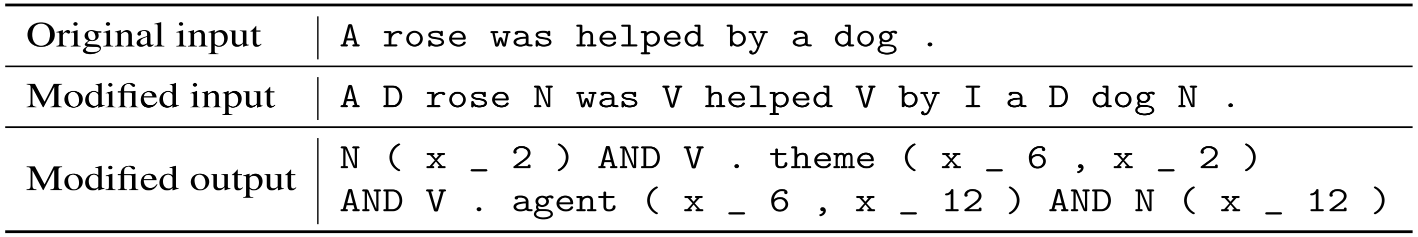
With data augmentation (1)

Table 4.4: Augmented inputs from the WP algorithm for COGS.



	Laughed	Giggled	Smiled
Sailor	The sailor laughed. (Original)	The sailor giggled.	The sailor smiled.
Soldier	The soldier laughed.	The soldier giggled.	The soldier smiled.
Boat	The boat laughed.	The boat giggled.	The boat smiled.

Table 4.5: Modified input and output representation on Word Permutation algorithm



Original input	A rose was helped by a dog .
Modified input	A D rose N was V helped V by I a D dog N .
Modified output	N (x _ 2) AND V . theme (x _ 6 , x _ 2) AND V . agent (x _ 6 , x _ 12) AND N (x _ 12)

With data augmentation (2)

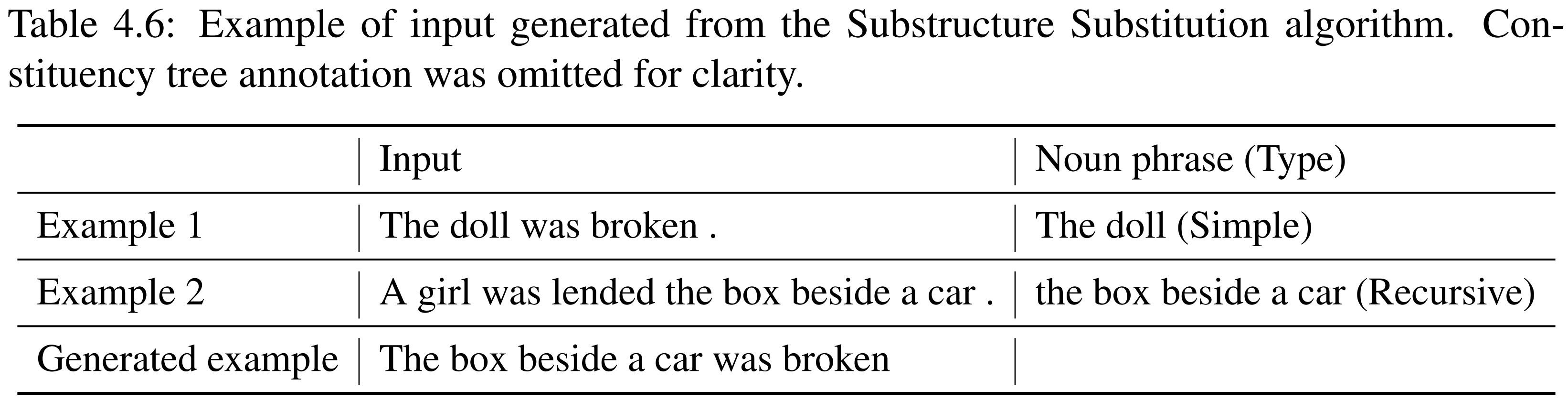


Table 4.6: Example of input generated from the Substructure Substitution algorithm. Constituency tree annotation was omitted for clarity.

	Input	Noun phrase (Type)
Example 1	The doll was broken .	The doll (Simple)
Example 2	A girl was lended the box beside a car .	the box beside a car (Recursive)
Generated example	The box beside a car was broken	

With data augmentation (3)

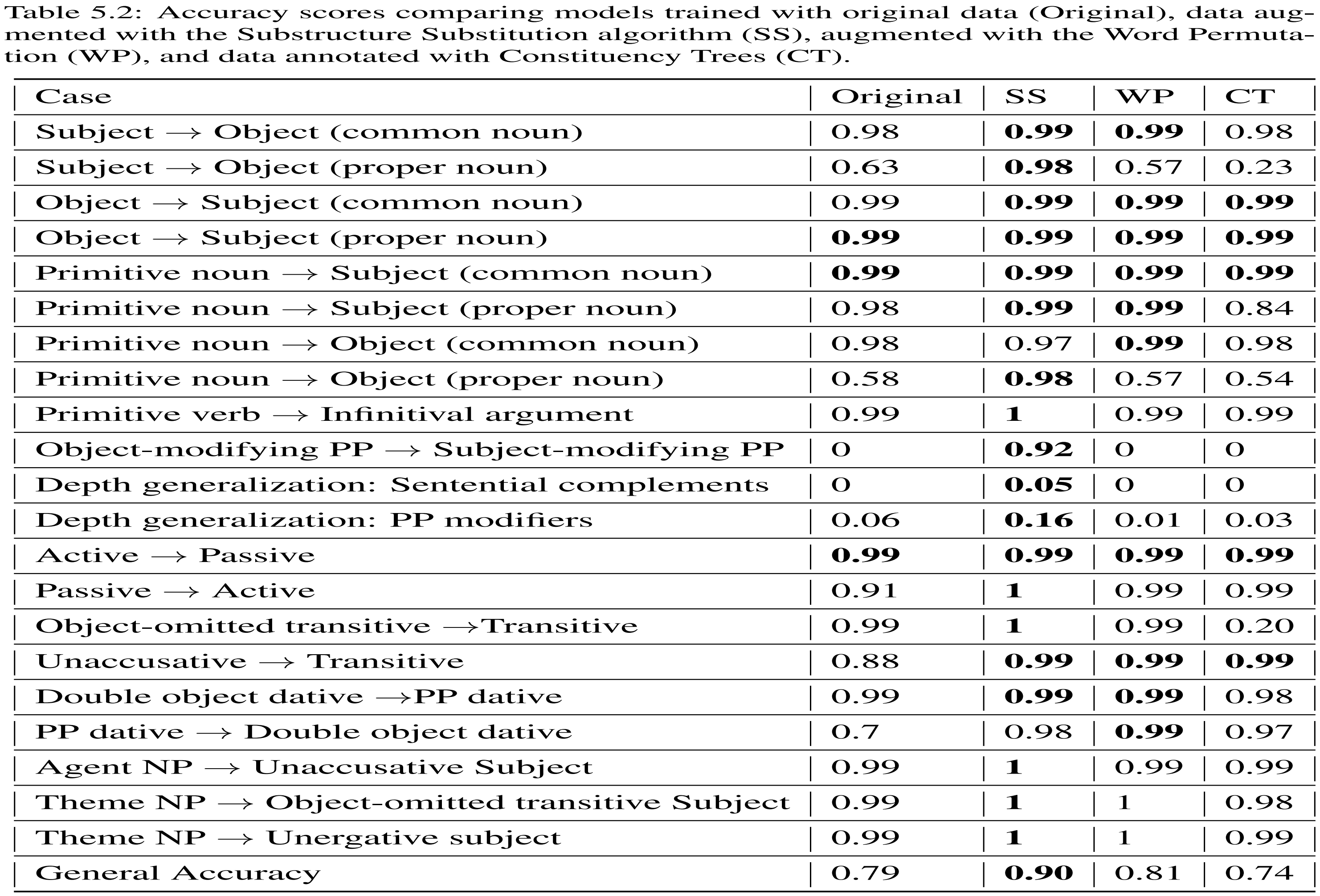
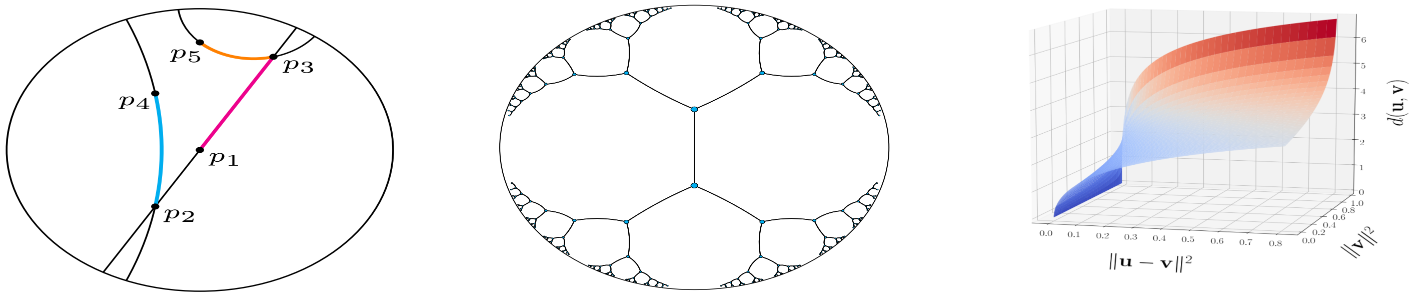


Table 5.2: Accuracy scores comparing models trained with original data (Original), data augmented with the Substructure Substitution algorithm (SS), augmented with the Word Permutation (WP), and data annotated with Constituency Trees (CT).

Case	Original	SS	WP	CT
Subject → Object (common noun)	0.98	**0.99**	**0.99**	0.98
Subject → Object (proper noun)	0.63	**0.98**	0.57	0.23
Object → Subject (common noun)	0.99	**0.99**	**0.99**	**0.99**
Object → Subject (proper noun)	**0.99**	**0.99**	**0.99**	**0.99**
Primitive noun → Subject (common noun)	**0.99**	**0.99**	**0.99**	**0.99**
Primitive noun → Subject (proper noun)	0.98	**0.99**	**0.99**	0.84
Primitive noun → Object (common noun)	0.98	0.97	**0.99**	0.98
Primitive noun → Object (proper noun)	0.58	**0.98**	0.57	0.54
Primitive verb → Infinitival argument	0.99	**1**	0.99	0.99
Object-modifying PP → Subject-modifying PP	0	**0.92**	0	0
Depth generalization: Sentential complements	0	**0.05**	0	0
Depth generalization: PP modifiers	0.06	**0.16**	0.01	0.03
Active → Passive	**0.99**	**0.99**	**0.99**	**0.99**
Passive → Active	0.91	**1**	0.99	0.99
Object-omitted transitive →Transitive	0.99	**1**	0.99	0.20
Unaccusative → Transitive	0.88	**0.99**	**0.99**	**0.99**
Double object dative →PP dative	0.99	**0.99**	**0.99**	0.98
PP dative → Double object dative	0.7	0.98	**0.99**	0.97
Agent NP → Unaccusative Subject	0.99	**1**	0.99	0.99
Theme NP → Object-omitted transitive Subject	0.99	**1**	1	0.98
Theme NP → Unergative subject	0.99	**1**	1	0.99
General Accuracy	0.79	**0.90**	0.81	0.74

Hyperbolic embeddings (1)



(a) Geodesics of the Poincaré disk (b) Embedding of a tree in $\mathcal{B}^2$ (c) Growth of Poincaré distance

Figure 1: (a) Due to the negative curvature of $\mathcal{B}$, the distance of points increases exponentially (relative to their Euclidean distance) the closer they are to the boundary. (c) Growth of the Poincaré distance $d(\boldsymbol{u}, \boldsymbol{v})$ relative to the Euclidean distance and the norm of $\boldsymbol{v}$ (for fixed $\|\boldsymbol{u}\| = 0.9$). (b) Embedding of a regular tree in $\mathcal{B}^2$ such that all connected nodes are spaced equally far apart (i.e., all black line segments have identical hyperbolic length).

Considering a target token t_k with its corresponding global hyperbolic embedding h_k^t, and its Euclidean embedding $e_k^{t,l-1}$ from the previous decoder layer, its self-attention score to another target token t_j is given by:

$$A^l(t_k, t_j) = softmax(\alpha A_h(h_k^t, h_j^t) + A_e^l(e_k^{t,l-1}, e_j^{t,l-1})) \qquad (6)$$

Hyperbolic embeddings (2)

- Résultats prometteurs avec une légère amélioration en combinant les deux types de plongement (euclidien et hyperbolique)
  - Les poids d’attention hyperbolique sont utilisés pour moyenner les plongements euclidiens (attention euclidienne utilisée par ailleurs)
  - Surcoût non négligeable