Functorial Language Models
(Work In Progress)

Alexis Toumi
University of Oxford
Cambridge Quantum Computing Ltd.
`alexis@toumi.email`

Alex Koziell-Pipe
`alex.koziellpipe@gmail.com`

Introduction

We introduce functorial language models: a principled way to compute probability distributions over word sequences given a monoidal functor from grammar to meaning. This yields a method for training categorical compositional distributional (DisCoCat) models on raw text data. We provide a proof-of-concept implementation in DisCoPy (de Felice et al., 2020), see appendix A.

Language models, i.e. probability distributions over word sequences, are a cornerstone of natural language processing and information retrieval (Ponte and Croft, 1998). Neural models (Bengio et al., 2003), where these distributions are learnt by a neural network, are now built in everyday tools such as virtual assistants, automatic translation, etc. However effective they may be, neural models are not compositional in the sense that they do not take the grammatical structure of sentences into account, at least not explicitly. This severely limits our ability to interpret them: if we open the black box, we only see matrices of weights.

On the other hand, categorical compositional distributional (DisCoCat) models (Clark et al., 2008, 2010) use grammatical structure to compose the distributional meaning of words together into a meaning for the sentence. Grammar is explicitly represented as string diagrams, which allow formal reasoning about natural language semantics, for example analysing ambiguity (Kartsaklis et al., 2013, 2014; Piedeleu et al., 2015) and entailment (Sadrzadeh et al., 2018a; de Felice et al., 2019).

However, despite emprical validation on small-scale examples (Grefenstette and Sadrzadeh, 2011), applying DisCoCat models at a large scale to real-world text data appears like a far-away goal. The challenge is at least two-fold: 1) we need a robust way to predict the grammatical structure of a given word sequence, 2) we need an efficient way to learn the meaning of each word, given the grammatical structures in which it occurs. The first point requires a probabilistic grammar: it is not enough to know whether a sequence is grammatical, we need a probability distribution over its possible parsings.

It has also been argued that a robust model ought to be incremental (Sadrzadeh et al., 2018b): scanning through the given word sequence left-to-right and updating its prediction for what comes next. This plays a crucial role in the context of informal language and dialogues (Purver et al., 2006). Previous work by the first author and collaborators (Shiebler et al., 2020) characterise these incremental, probabilistic parsers in terms of a functor from formal grammars to automata.

The second challenge, i.e. how to use grammar for learning the meaning of words, is essential if we want DisCoCat to work as a standalone model for natural language. Indeed, the first experiments (Grefenstette and Sadrzadeh, 2011) assumed the vector embedding for nouns to be given by some other means, such as co-occurence counts or neural language models. On the other hand, in more recent experiments on quantum hardware (Meichanetzidis et al., 2020a,b; Coecke et al., 2020) the meaning of words is learnt directly from the training data of a supervised question-answering task. We coin the term *functorial learning* for this approach, as it can be understood as learning a functor from data.

We give a formal definition of this functorial approach and define *functorial language models* by composition with a probabilistic grammar. Concretely, we train a DisCoCat model to predict which word is missing in a sentence with a hole. We argue that this captures precisely the idea of distributional compositionality: extending Firth's principle, we shall know a word by the company it keeps and by the grammatical structure in which it occurs.

1 Functorial learning for DisCoCat

DisCoCat models have a one-sentence definition: they are rigid monoidal functors from the category generated by a pregroup grammar to vector spaces[1] and linear maps. Let's unpack this definition and repack it in a format suitable for computation.

Fix a set of words V, called the vocabulary, and a set of basic types X. A rule is a pair of sequences of words and types $r \in (V + X)^\star \times (V + X)^\star$, where $\times$, $+$ and * denote respectively the Cartesian product, the disjoint union and the free monoid $X^\star = \coprod_{n \in \mathbb{N}} X^n$ with unit 1 the empty string. A grammar $G = (V, X, R, s)$ is given by a set R of such rules together with a distinguished $s \in X$ called the sentence type. This defines a signature in the sense of (Selinger, 2010) which generates a free monoidal category $\mathbf{G}$. The objects are sequences of words and types, the arrows are progressive[2] planar[3] string diagrams with rules as boxes. The language of G is defined as the sequences $x \in V^\star$ such that there is a diagram $g : x \to s$ in $\mathbf{G}$: its grammatical structure.

This definition merely reformulates the notion of semi-Thue system (Thue, 1914) in the language of monoidal category theory. (Post, 1947) and (Markov, 1947) independently proved the undecidability of the parsing problem, i.e. given G and $x \in V^\star$ decide whether there exists $g : x \to s$ in $\mathbf{G}$. (Chomsky, 1957) then put the equivalent notion of unrestricted grammar at the bottom of his well-known hierarchy. In parallel, (Lambek, 1958) defined his calculus in terms of closed monoidal categories, taking inspiration from the categorial grammars of (Adjukiewicz, 1935) and (Bar-Hillel, 1953). Half a century later, pregroup grammars (Lambek, 1999, 2001, 2008) simplified this calculus by going from closed to rigid monoidal categories. Pregroup grammars are at the basis of the original DisCoCat model of (Clark et al., 2008).

Definition 1.1. *A pregroup grammar has types $X \times \mathbb{Z}$ where for $n \in \mathbb{N}$ the types $(x, -n)$ and $(x, +n)$ are written $x^{l \dots l}$ and $x^{r \dots r}$ respectively and are called iterated left and right adjoints. The rules are of two kinds: cups and triangles. Cups have the shape $(x, z)(x, z + 1) \to 1$ for $z \in \mathbb{Z}$, i.e. they cancel a basic type with its right adjoint. Triangles have the shape $w \to t$ for a word $w \in V$ and a type $t \in (X \times \mathbb{Z})^\star$. Note that we draw words as the labels of triangle boxes rather than as input wires.*

A pregroup grammar $G = (V, X, R, s)$ defines a rigid signature: the generating objects are given by words and types $V + X$, the generating arrows are given by dictionary entries $w \to t \in R$. Thus, it generates a free rigid monoidal category $\mathbf{G}$ where the object are sequences of words and types with iterated adjoints, the arrows are planar string diagrams with triangles as boxes. Cups are given by the rigid structure of $\mathbf{G}$, they are drawn as bent wires. Again, the language of G is defined as $\{x \in V^\star \mid \exists\, g : x \to s \in \mathbf{G}\}$.

The main distinction between Chomsky's phrase structure grammars and the categorial grammar tradition of Adjukiewicz, Bar-Hillel and Lambek is that in the latter "all the grammar is in the lexicon". The only language-dependent rules are dictionary entries of the form $r : w \to t$ for a word $w \in V$ and a type $t \in \mathrm{Ob}(\mathbf{G})$. They are drawn as triangle boxes. All the grammatical rules come from the structure of the category $\mathbf{G}$, e.g. the cups of pregroups from the rigid structure. This has both conceptual and computational advantages. Conceptually, it makes categorial grammars universal: the same rules apply to all languages, only the dictionaries change. Computationally, these universal rules have a canonical semantics, thus it is enough to define the meaning of each dictionary entry.

Definition 1.2. *A DisCoCat model is a rigid monoidal functor $F : \mathbf{G} \to \mathbf{Vect}$ from the category generated by a pregroup grammar. On objects F is defined by a mapping $F_0 : X \to \mathbb{N}$ which sends each basic type to the dimension of a vector space. The image of word types $w \in V$ is the monoidal unit $F(w) = 1$, the image of complex types $t = (x_1, z_1) \dots (x_n, z_n)$ is the product of the dimensions of their basic types $F(t) = F_0(x_1) \dots F_0(x_n)$. On arrows F is defined by a mapping $F_1 \in \coprod_{(w,t) \in R} \mathbb{R}^{F(t)}$ that sends each dictionary entry $(w, t) \in R$ to a vector $F(w, t) \in \mathbb{R}^n$ for $n = F(t)$ the dimension of its type. Cups are sent to the rigid monoidal structure of* **Vect**. *The meaning of a sentence $g : x \to s$ is given by the vector $F(g) \in \mathbb{R}^m$ for $m = F(s)$ the dimension of the sentence type. When $F(s) = 1$, the meaning of sentences is given by real-valued scalars that can encode either truth value or likelihood.*

Remark 1.3. *DisCoCat models have been generalised both in their domain and codomain: one can vary the grammar category as in (Coecke et al.,*

[1] We only consider finite dimensional vector spaces.
[2] A diagram is progressive when its wires go monotonously top to bottom, i.e. they do not bend up or down.
[3] A diagram is planar when its wires do not cross.

2013) or the semantic category as in (Piedeleu et al., 2015; Coecke et al., 2018b; Delpeuch, 2019). While we stick to the original definition, our proposal can apply to any model, so long as the meaning of words lives in some vector space, regardless of how these word vectors are composed together.

We fix the map $F_0 : X \to \mathbb{N}$ and consider it as the hyper-parameters of the model. All the data defining the DisCoCat model is contained in the finite set of vectors $F_1 \in \coprod_{(w,t)\in R} \mathbb{R}^{F(t)}$. Thus, we may consider these $D = \sum_{(w,t)\in R} F(t)$ parameters as a machine learning landscape. The idea of functorial learning is to take a training set of pairs $\mathcal{X} \subseteq Ar(\mathbf{G}) \times Ar(\mathbf{Vect})$ of diagrams $d \in Ar(\mathbf{G})$ and vectors $y \in Ar(\mathbf{Vect})$, and learn a functor F such that $F(d) = y$. In practice, this exact functor may not exist thus we fix a loss $L : \coprod_{n\in\mathbb{N}} \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ and a regularisation $R : \mathbb{R}^D \to \mathbb{R}^+$ then using gradient-based methods we approximate:

$$\underset{F:\mathbf{G}\to\mathbf{Vect}}{\arg\min} \quad R(F_1) + \sum_{(d,y)\in\mathcal{X}} L\big(F(d), y\big)$$

Example 1.4. *Fix a type $q \in X$ for yes-no questions with $F(q) = 1$, i.e. the meaning of a question $d : w_1 \dots w_n \to q$ is a scalar $F(d) \in \mathbb{R}$. Take a training set of such questions together with their answer in $\{0, 1\} \subseteq \mathbb{R}$. Then the functor F may be seen as a DisCoCat model for question-answering, see (de Felice et al., 2019). This model has been deployed on quantum hardware, see (Meichanetzidis et al., 2020a,b; Coecke et al., 2020).*

2 Functorial language models

The definition of DisCoCat models can be recast in terms of encoding matrices, first introduced in (Toumi, 2018; Coecke et al., 2018a). The set of meaning vectors $F_1 \in \coprod_{(w,t)\in R} \mathbb{R}^{F(t)}$ can be packed into a set of matrices $E_t : |V_t| \to F_0(t)$ indexed by grammatical types $t \in (X \times \mathbb{Z})^\star$, where $V_t \subseteq V$ is the set of words $w \in V$ with $(w, t) \in R$. If we compose the matrix E_t with the one-hot vector for a word $w : 1 \to |V_t|$, we get its image under the functor $F(w, t) = w \fatsemi E_t$. In the other direction, if we compose a meaning (co)vector $v : F_0(t) \to 1$ with E_t, we get a (co)vector $E_t \fatsemi v : |V_t| \to 1$ which gives its inner product with the words in V_t. The key insight of our functorial language models is to take $\text{softmax}(E_t \fatsemi v)$ as the probability distribution over words in V_t given a meaning $v : F_0(t) \to 1$.

Remark 2.1. *softmax is not a linear map, hence technically we cannot draw it as a box in a* **Vect***-valued diagram. We can however draw it as a bubble, see (Toumi et al., 2021). We can then consider functors which send bubbles to softmax.*

Fix a corpus of grammatical sentences $\mathcal{X} \subseteq \coprod_{x\in V^\star} \mathbf{G}(x, s)$ and assume $F(s) = 1$, i.e. the meaning of sentences are scalars. We generate a training set from this by taking each sentence $d : w_1 \dots w_n \to s$, removing one dictionary entry $w_i \to t_i$ by replacing it with the identity $t_i \to t_i$. Thus, we get a diagram $d_i : t_i \to s$ and take the one-hot vector for the word w_i as target label. Intuitively, given the diagram of a sentence with a hole, we want to predict what word is missing.

This does not yet define a language model in the usual sense, indeed we are assuming that text data comes annotated with grammatical structure. That is, we are computing the conditional distribution $P(w|d)$ over words $w \in V_t$ of type $t \in \text{Ob}(\mathbf{G})$ given the grammatical structure $d : t \to s$ in which they occur. In order to get a language model, we need to compose this conditional with a distribution $P(d|w_1 \dots w_n)$ over diagrams given word sequences: a probabilistic grammar. Learning this distribution from data is called probabilistic grammar induction, which (Shiebler et al., 2020) formulate in terms of monoidal categories. In future work, we plan to train our functorial language model end-to-end on raw text data, i.e. learning both the probabilistic grammar and the DisCoCat model at once.

3 Implementation

We implemented a proof-of-concept in DisCoPy (de Felice et al., 2020) and used jax (Bradbury et al., 2020) for automatic differentiation and just-in-time compilation on GPU. We initialise the encoding matrices at random and use singular value decomposition to obtain word vectors that are close to orthogonal. We used cross-entropy loss, a weighted sum of l1 and l2 (1e−1 and 5e−2 resp.) regularisation and Adam optimization (Kingma and Ba, 2017) (learning rate = 5e−2). We set the hyper-parameters to $F(s) = 1$ and $F(n) = 7$. After training on a subset of hand-crafted data (~100 sentences with three distinct grammatical structures), the model was able to infer the missing word in previously unseen sentences with accuracy $> .75$, see appendix A. The notebook can be found on DisCoPy's GitHub.

References

Adjukiewicz, K. (1935). Die syntaktische Konnexität. *Studia Philosophica*.

Bar-Hillel, Y. (1953). A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29(1):47.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., and Wanderman-Milne, S. (2020). JAX: Composable transformations of Python+ NumPy programs, 2018. *URL http://github. com/google/jax*, 4:16.

Chomsky, N. (1957). *Syntactic Structures*. Mouton and Co., The Hague.

Clark, S., Coecke, B., and Sadrzadeh, M. (2008). A Compositional Distributional Model of Meaning. In *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, pages 133–140.

Clark, S., Coecke, B., and Sadrzadeh, M. (2010). Mathematical foundations for a compositional distributional model of meaning. In van Benthem, J., Moortgat, M., and Buszkowski, W., editors, *A Festschrift for Jim Lambek*, volume 36 of *Linguistic Analysis*, pages 345–384.

Coecke, B., de Felice, G., Marsden, D., and Toumi, A. (2018a). Towards Compositional Distributional Discourse Analysis. *Electronic Proceedings in Theoretical Computer Science*, 283:1–12.

Coecke, B., de Felice, G., Meichanetzidis, K., and Toumi, A. (2020). Foundations for Near-Term Quantum Natural Language Processing. *arXiv:2012.03755 [quant-ph]*.

Coecke, B., Genovese, F., Lewis, M., Marsden, D., and Toumi, A. (2018b). Generalized relations in linguistics & cognition. *Theoretical Computer Science*.

Coecke, B., Grefenstette, E., and Sadrzadeh, M. (2013). Lambek vs. Lambek: Functorial Vector Space Semantics and String Diagrams for Lambek Calculus. *ArXiv e-prints*.

de Felice, G., Meichanetzidis, K., and Toumi, A. (2019). Functorial Question Answering. *arXiv:1905.07408 [cs, math]*.

de Felice, G., Toumi, A., and Coecke, B. (2020). DisCoPy: Monoidal Categories in Python. *arXiv:2005.02975 [math]*.

Delpeuch, A. (2019). Autonomization of Monoidal Categories. *arXiv:1411.3827 [cs, math]*.

Grefenstette, E. and Sadrzadeh, M. (2011). Experimental Support for a Categorical Compositional Distributional Model of Meaning. *arXiv:1106.4058 [cs, math]*, pages 1394–1404.

Kartsaklis, D., Sadrzadeh, M., and Pulman, S. (2013). Separating Disambiguation from Composition in Distributional Semantics. page 10.

Kartsaklis, D., Sadrzadeh, M., Pulman, S., and Coecke, B. (2014). Reasoning about Meaning in Natural Language with Compact Closed Categories and Frobenius Algebras. *CoRR*, abs/1401.5980.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Lambek, J. (1958). The Mathematics of Sentence Structure. *The American Mathematical Monthly*, 65(3):154–170.

Lambek, J. (1999). Type Grammar Revisited. In Lecomte, A., Lamarche, F., and Perrier, G., editors, *Logical Aspects of Computational Linguistics*, pages 1–27, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lambek, J. (2001). Type Grammars as Pregroups. *Grammars*, 4:21–39.

Lambek, J. (2008). *From Word to Sentence: A Computational Algebraic Approach to Grammar*. Open Access Publications. Polimetrica.

Markov, A. (1947). On certain insoluble problems concerning matrices. In *Doklady Akad. Nauk SSSR*, volume 57, pages 539–542.

Meichanetzidis, K., Gogioso, S., De Felice, G., Chiappori, N., Toumi, A., and Coecke, B. (2020a). Quantum Natural Language Processing on Near-Term Quantum Computers. *arXiv:2005.04147 [quant-ph]*.

Meichanetzidis, K., Toumi, A., de Felice, G., and Coecke, B. (2020b). Grammar-Aware Question-Answering on Quantum Computers. *arXiv:2012.03756 [quant-ph]*.

Piedeleu, R., Kartsaklis, D., Coecke, B., and Sadrzadeh, M. (2015). Open System Categorical Quantum Semantics in Natural Language Processing. *CoRR*, abs/1502.00831.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281.

Post, E. L. (1947). Recursive Unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12(1):1–11.

Purver, M., Cann, R., and Kempson, R. (2006). Grammars as Parsers: Meeting the Dialogue Challenge. *Research on Language and Computation*, 4(2-3):289–326.

Sadrzadeh, M., Kartsaklis, D., and Balkır, E. (2018a). Sentence entailment in compositional distributional semantics. *Annals of Mathematics and Artificial Intelligence*, 82(4):189–218.

Sadrzadeh, M., Purver, M., Hough, J., and Kempson, R. (2018b). Exploring Semantic Incrementality with Dynamic Syntax and Vector Space Semantics. *arXiv:1811.00614 [cs]*.

Selinger, P. (2010). A Survey of Graphical Languages for Monoidal Categories. *New Structures for Physics*, pages 289–355.

Shiebler, D., Toumi, A., and Sadrzadeh, M. (2020). Incremental Monoidal Grammars. *arXiv:2001.02296 [cs]*.

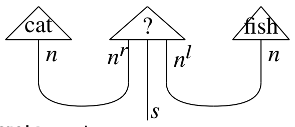
Thue, A. (1914). *Probleme Über Veränderungen von Zeichenreihen Nach Gegebenen Regeln*. na.

Toumi, A. (2018). Categorical compositional dIstributional questions, answers & discourse analysis. Technical report, University of Oxford.

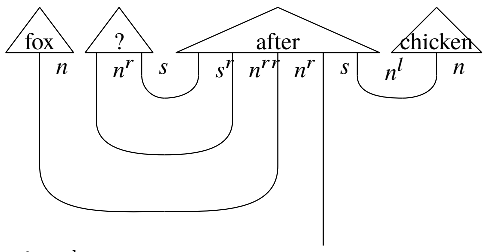
Toumi, A., Yeung, R., and de Felice, G. (2021). Diagrammatic Differentiation for Quantum Machine Learning. *arXiv:2103.07960 [quant-ph]*.

A Experiment

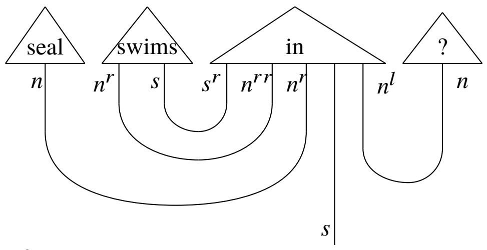
We reproduce our hand-crafted dataset together with some sample predictions from our model. In the following cherry-picked examples, the prediction is correct in the first three cases and incorrect in the last one. This failure can be explained by the fact the word "krill" only appears once in the training set.



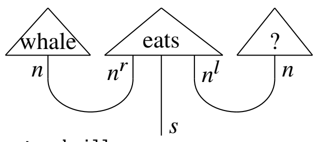
Target: `eats`
Prediction: `eats (0.60), bites (0.35), flees (0.05)`



Target: `chases`
Prediction: `chases (1.00)`



Target: `water`
Prediction: `water (0.97), dog (0.02)`



Target: `krill`
Prediction: `cheese (0.53), fish (0.17), grain (0.10)`

```
?  runs after mouse (cat)
?  runs on land (seal)
dog chases after ?  (fox)
?  barks at fox (dog)
whale eats ?  (krill)
fox flees ?  (dog)
dog barks at ?  (fox)
seal swims in ?  (water)
chicken runs on ?  (land)
dog bites ?  (bone)
fox chases after ?  (chicken)
cat ?  fish (eats)
dog ?  cat (bites)
chicken ?  fox (flees)
mouse ?  (squeaks)
cat ?  (meows)
fish ?  (swims)
fox ?  after chicken (chases)
fish ?  in water (swims)
cat chases ?  mouse (after)
mouse runs ?  land (on)
cat runs ?  land (on)

dog chases ?  fox (after)
```

Figure 1: Testing set (23 sentences)

```
?  runs after chicken (fox)
mouse runs on ?  (land)
seal eats ?  (fish)
cat eats ?  (fish)
?  runs after fox (dog)
dog runs after ?  (fox)
dog runs on ?  (land)
?  swims (seal)
?  squeaks (mouse)
chicken eats ?  (grain)
fox chases ?  (chicken)
seal runs on ?  (land)
cat chases ?  (mouse)
cat runs on ?  (land)
?  runs after cat (dog)
cat runs after ?  (mouse)
whale swims in ?  (water)
mouse flees ?  (cat)
dog eats ?  (bone)
fox runs on ?  (land)
?  chases after chicken (fox)
?  swims in water (seal)
fish swims in ?  (water)
cat chases after ?  (mouse)
fox runs after ?  (chicken)
fox bites ?  (chicken)
?  barks at cat (dog)
?  chases (fox)
mouse bites ?  (cheese)
?  chases after cat (dog)
?  barks (dog)
?  chases after fox (dog)
?  chases after mouse (cat)
?  clucks (chicken)
mouse eats ?  (cheese)
cat bites ?  (fish)
chicken flees ?  (fox)
fox eats ?  (chicken)
?  meows (cat)
dog chases ?  (fox)
?  runs (seal)
cat flees ?  (dog)
seal ?  fish (eats)
mouse ?  cheese (eats)
dog ?  fox (bites)
whale ?  krill (eats)
dog ?  bone (eats)
fox ?  chicken (eats)
chicken ?  grain (eats)
cat ?  dog (flees)
mouse ?  cat (flees)
cat ?  mouse (bites)
fox ?  dog (flees)
chicken ?  (clucks)
dog ?  (barks)
chicken ?  on land (runs)
whale ?  (swims)
mouse ?  on land (runs)
dog ?  at cat (barks)
seal ?  on land (runs)
cat ?  on land (runs)
whale ?  in water (swims)
dog ?  at fox (barks)
dog ?  on land (runs)
dog ?  after fox (chases)
fox ?  (chases)
seal ?  (runs)
seal ?  in water (swims)
cat ?  after mouse (chases)
fox ?  on land (runs)
dog ?  after cat (chases)
seal runs ?  land (on)
seal swims ?  water (in)
fish swims ?  water (in)
fox runs ?  land (on)
fox chases ?  chicken (after)
fox runs ?  chicken (after)
whale swims ?  water (in)
dog runs ?  cat (after)
cat runs ?  mouse (after)
dog runs ?  fox (after)
dog barks ?  fox (at)
dog barks ?  cat (at)
dog runs ?  land (on)
chicken runs ?  land (on)

dog chases ?  cat (after)
```

Figure 2: Training set (86 sentences)