Sentence embeddings and their relation with sentence structures

Antoine Simoulin

Université Paris Cité

Sciences Mathématiques de Paris Centre (ED 386)

Laboratoire de Linguistique Formelle (LLF)

Sentence embeddings and their relation with sentence structures

Par Antoine Simoulin

Thèse de doctorat d'informatique

Dirigée par Benoit Crabbé

Présentée et soutenue publiquement le 7 juillet 2022

Devant un jury composé de :

Claire Gardent, directrice de recherche, CNRS et Université de Lorraine, rapporteuse
Éric Gaussier, professeur HDR, Université Grenoble Alpes, rapporteur
Rachel Bawden, chargée de recherche, Inria, examinatrice
Loïc Barrault, maitre de conférence, Le Mans Université, examinateur
Benoit Crabbé, professeur HDR, Université Paris Cité, directeur de thèse
Nicolas Brunel, professeur, ENSIIE et Laboratoire de Mathématiques et Modélisation d'Évry, membre invité du jury

Titre : Plongements de phrases et leurs relations avec les structures de phrases

Résumé (court) : Historiquement, la modélisation du langage humain suppose que les phrases ont une structure symbolique et que cette structure permet d'en calculer le sens par composition. Ces dernières années, les modèles d'apprentissage profond sont parvenus à traiter automatiquement des tâches sans s'appuyer sur une structure explicite du langage, remettant ainsi en question cette hypothèse fondamentale. Cette thèse cherche ainsi à mieux identifier le rôle de la structure lors de la modélisation du langage par des modèles d'apprentissage profonds. Elle se place dans le cadre spécifique de la construction de plongements de phrases—des représentations sémantiques basées sur des vecteurs—par des réseaux de neurones profonds. Dans un premier temps, on étudie l'intégration de biais linguistiques dans les architectures de réseaux neuronaux, pour contraindre leur séquence de composition selon une structure traditionnelle, en arbres. Dans un second temps, on relâche ces contraintes pour analyser les structures latentes induites par ces réseaux neuronaux. Dans les deux cas, on analyse les propriétés de composition des modèles ainsi que les propriétés sémantiques des plongements. La thèse s'ouvre sur un état de l'art présentant les principales méthodes de représentation du sens des phrases, qu'elles soient symboliques, ou basées sur des méthodes d'apprentissage profond. La deuxième partie propose plusieurs expériences introduisant des biais linguistiques dans les architectures des réseaux de neurones pour construire des plongements de phrases. Le premier chapitre combine explicitement plusieurs structures de phrases pour construire des représentations sémantiques. Le deuxième chapitre apprend conjointement des structures symboliques et des représentations vectorielles. Le troisième chapitre introduit un cadre formel pour les transformers selon une structure de graphes. Finalement, le quatrième chapitre étudie l'impact de la structure vis-à-vis de la capacité de généralisation et de compositions des modèles. La thèse se termine par une mise en concurrence de ces approches avec des méthodes de passage à l'échelle. On cherche à y discuter les tendances actuelles qui privilégient des modèles plus gros, plus facilement parallélisables et entraînés sur plus de données, aux dépens de modélisations plus fines. Les deux chapitres de cette partie relatent l'entraînement de larges modèles de traitement automatique du langage et comparent ces approches avec celles développées dans la deuxième partie d'un point de vue qualitatif et quantitatif.

Résumé (long) : Historiquement, la modélisation du langage humain suppose que les phrases ont une structure symbolique et que cette structure permet d'en calculer le sens par composition. Ces dernières années, les modèles d'apprentissage profond sont parvenus à traiter automatiquement des tâches sans s'appuyer sur une structure explicite du langage, remettant ainsi en question cette hypothèse fondamentale. Cette thèse cherche ainsi à mieux identifier le rôle de la structure lors de la modélisation du langage

par des modèles d'apprentissage profonds. Elle se place dans le cadre spécifique de la construction de plongements de phrases—des représentations sémantiques basées sur des vecteurs—par des réseaux de neurones profonds. Dans un premier temps, nous étudions l'intégration de biais linguistiques dans les architectures de réseaux neuronaux, pour contraindre leur séquence de composition selon une structure traditionnelle, en arbres. Dans un second temps, nous relâchons ces contraintes pour analyser les structures latentes induites par ces réseaux neuronaux. Dans les deux cas, nous analysons les propriétés de composition des modèles ainsi que les propriétés sémantiques des plongements. La thèse est financée par Quantmetry—un cabinet de conseil français pionnier dans le domaine de l'intelligence artificielle—qui travaille sur des projets d'IA de bout en bout, de la stratégie à l'industrialisation. Ainsi, les travaux sont motivés par un désir de proposer des contributions actionnables dans le milieu industriel et des outils efficaces pour des applications concrètes. Une grande partie de ce travail est donc publiée sous forme d'outils et de contributions en accès libre.

La thèse s'ouvre sur un état de l'art présentant les principales méthodes de représentation du sens des phrases, qu'elles soient symboliques, ou basées sur des méthodes d'apprentissage profond.

La deuxième partie propose plusieurs expériences introduisant des biais linguistiques dans les architectures des réseaux de neurones pour construire des plongements de phrases. Cette partie comporte quatre chapitres.

Le premier chapitre combine explicitement plusieurs structures de phrases pour construire des représentations sémantiques. Nous supposons que la signification d'une phrase est une fonction des aspects syntaxiques et sémantiques. À cet égard, nous proposons une méthode auto-supervisée qui construit des plongements de phrases à partir de la combinaison de diverses structures syntaxiques. La nouveauté consiste à proposer une approche multi-vue qui apprend conjointement des modèles structurés en induisant une interaction explicite entre eux pendant la phase d'apprentissage. Nous pré-entraînons plusieurs modèles en utilisant un objectif contrastif avec un corpus de 40 millions de phrases. Nous évaluons ensuite nos modèles sur des ressources d'évaluation des plongements de phrases et obtenons des résultats à l'état de l'art. En particulier sur des tâches qui devraient, par hypothèse, être plus sensibles à la structure des phrases.

Le deuxième chapitre apprend conjointement des structures symboliques et des représentations vectorielles. Nous utilisons des réseaux neuronaux structurés en arbre, qui encodent naturellement la structure du langage. Pour chaque phrase, le réseau encode les unités de texte en suivant un arbre syntaxique, en partant des feuilles jusqu'à la racine. Cependant, ces modèles souffrent de contraintes pratiques qui limitent leur application. En particulier, les modèles structurés nécessitent non seulement du texte brut en entrée mais aussi la structure de la phrase sous la forme d'un arbre. Une telle structure nécessite des annotations dans le cadre supervisé. Nous formulons un nouveau modèle structuré en arbre qui apprend sa fonction de composition en même temps que sa structure. Le modèle comprend deux modules, un analyseur de graphe biaffine et un Tree-LSTM. Les fonctions d'analyse syntaxique et de composition sont explicitement connectées et, par conséquent,

apprises conjointement. La méthode diffère d'autres approches car la représentation n'est pas calculée à partir d'une forêt entière d'arbres potentiels. De plus, l'apprentissage du modèle ne nécessite pas de supervision directe pour la structure. Le modèle est plus performant que les modèles à base d'arbres reposant sur des structures extrinsèques. Dans certaines configurations, il est même compétitif avec Bert.

Le troisième chapitre introduit un cadre formel pour les transformers selon une structure de graphes. Les architectures de transformers ont gagné en popularité au sein de la communauté. Contrairement aux modèles basés sur des arbres, ils n'ont pas besoin de données annotées pour être entraînés. D'autre part, comme le suggèrent de nombreux résultats, ces nouveaux modèles acquièrent une forme de structure hiérarchique. Les transformers transforment simultanément l'ensemble des représentations des *tokens*—les unités lexicales d'une phrase—selon un nombre fixe de couches. Néanmoins, le rôle de ces couches et la façon dont elles traitent l'information ne sont pas entièrement compris. Nous formulons l'hypothèse que les couches ne codent pas spécifiquement des fonctions surfaciques, syntaxiques ou sémantiques mais plutôt que de telles informations émergent par l'application itérative des couches. Pour mieux étudier la transformation des représentations des *tokens* à travers les couches, nous proposons une variante du modèle Albert. A l'instar d'Albert, notre modèle partage ses poids entre l'ensemble des couches mais adapte aussi dynamiquement le nombre de couches appliquées à chaque *token*. Nous analysons le processus de transformation des *tokens* selon la profondeur du réseau. En particulier, nous étudions comment les itérations sont distribuées en fonction des types de dépendance des *tokens*. Nous montrons que les *tokens* ne nécessitent pas le même nombre d'itérations et que les *tokens* difficiles ou cruciaux pour la tâche nécessitent plus d'itérations.

Finalement, le quatrième chapitre étudie l'impact de la structure vis-à-vis de la capacité de généralisation et de compositions des modèles. Bien que les transformers améliorent les performances sur de nombreux *benchmarks*, ils présentent également certaines limites. En particulier en ce qui concerne leur capacité à généraliser en dehors de leur domaine d'entraînement et à apprendre des règles de composition élémentaires. En particulier, les modèles d'apprentissage profond ont du mal à généraliser à des séquences plus longues ou à des phrases présentant des niveaux de récusions plus profonds que ceux vus pendant l'entraînement. Pour faire suite à nos travaux sur l'intégration de la structure dans les architectures neuronales, nous cherchons à mieux caractériser le rôle de la structure dans les propriétés compositionnelles des modèles. Ce travail est actuellement en phase d'expérimentation. Nous construisons une méthode d'évaluation avec des expressions arithmétiques contenant des propriétés spécifiques. Nous entraînons différents modèles sur des sous-ensembles du jeu de données et observons comment les modèles généralisent en dehors de leur domaine. Nous comparons des modèles intégrants divers niveaux de contraintes structurelles : des modèles séquentiels, récursifs ou non structurés.

La thèse se termine par une mise en concurrence de ces approches avec des méthodes de passage à l'échelle. Dans cette seconde partie, on cherche à discuter les tendances actuelles qui privilégient des modèles plus gros, plus facilement parallélisables et entraînés sur plus de données, aux dépens de modélisations plus fines. Les deux chapitres de cette

partie relatent l'entraînement de larges modèles de traitement automatique du langage et comparent ces approches avec celles développées dans la deuxième partie d'un point de vue qualitatif et quantitatif.

Le premier chapitre s'interroge sur la taille des modèles. à première vue, il semble que le traitement actuel du langage naturel évolue vers des modèles de plus en plus gros aux dépens des subtilités de leur architecture. Cette tendance n'a pas directement profité aux modèles de plongements de phrases, car de nombreux encodeurs à base de transformers affichent des performances inférieures à l'état de l'art sur les *benchmarks* d'évaluation. Dans cette section, nous explorons comment nous pouvons faire évoluer les performances des encodeurs de phrases en adaptant leur pré-entraînement et en augmentant leur taille. Nous détaillons le développement, l'entraînement et le partage de modèles de plongements de phrases à l'état de l'art. Nous utilisons un objectif contrastif et entraînons les modèles sur un corpus d'un milliard de phrases.

Le second chapitre de cette partie s'intéresse à l'entrainement d'un modèle de langue incrémental en français. Ce type de modèle peut acquérir des compétences grammaticales très impressionnantes. Par exemple, GPT-2 génère un texte correct avec un accord au pluriel et à distance, et ce, en dépit toute connaissance linguistique préalable. Ces accords sont pourtant déterminés par des structures abstraites et pas seulement par l'ordre linéaire des mots. Plus largement, les modèles peuvent apprendre de nombreux motifs linguistiques (sujet-verbe, nom-adverbe, verbe-verbe) sans aucune information préalable sur la théorie linguistique. Au sein de notre laboratoire, nous avons dirigé le projet d'entraînement du premier grand modèle de langage incrémental en français. Nous avons obtenu une subvention de calcul pour le calculateur public français Jean Zay. Le modèle, équivalent à GPT-2 en anglais, contient plus d'un milliard de paramètres. Nous avons construit un corpus d'entraînement dédié et parallélisé l'entraînement entre plusieurs nœuds et unités de calcul. Nous avons publié le modèle en Open-Source pour la recherche et les applications commerciales.

En conclusion, nous avons étudié le rôle de la structure des modèles neuronaux pour composer les unités lexicales lors de la construction de plongements de phrases. Nos travaux ont abordé plusieurs problèmes critiques des plongements de phrases, notamment le manque de robustesse vis-à-vis de la généralisation hors du domaine, le manque de propriétés de compositions, la nécessité de vastes corpus d'entraînement ou la sur paramétrisation. Nos travaux apportent plusieurs contributions.

Tout d'abord, nous avons donné des éléments empiriques montrant que certaines structures de réseaux neuronaux sont plus appropriées pour capturer des types d'informations spécifiques. Nous avons ainsi observé empiriquement que la structure des modèles impacte la nature des informations accessibles dans les plongements de phrases. Nous avons ainsi confirmé notre hypothèse selon laquelle la combinaison de diverses structures devrait être plus robuste pour les tâches nécessitant des opérations de compositions subtiles.

Deuxièmement, nous avons proposé des architectures originales pour apprendre conjointement la structure de la phrase et la fonction de composition sémantique. Par conséquent, l'apprentissage du modèle ne nécessite pas la supervision d'un objectif d'analyse syntaxique.

Nous avons également étudié structures latentes apprises par les modèles transformers et proposé un cadre d'interprétation sous forme de réseaux de graphes. Nos expériences fournissent une nouvelle voie d'interprétation pour le rôle des couches dans les modèles de transformers profonds. Plutôt que d'extraire des caractéristiques spécifiques à chaque étape, les couches pourraient agir comme un processus itératif et convergent.

Troisièmement, nous avons adapté la méthode standard de pré-entraînement contrastive pour entraîner des modèles de transformers de grande taille sur un grand ensemble de données. Nous avons ainsi réussi à étendre leur pré-entraînement pour surpasser les approches précédentes. Toujours dans le cadre du passage à l'échelle, nous avons pré-entraîné une version française du modèle GPT.

Enfin, nous avons développé des ressources d'évaluation. Nous avons développé un jeu de données pour évaluer les propriétés de composition des modèles. En s'appuyant sur des expressions arithmétiques dont nous contrôlons les caractéristiques, nous évaluons des propriétés qui s'appliquent également à l'étude du langage humain. Nous avons également introduit un jeu de données d'évaluation pour les modèles de langue française.

Mots clefs : Traitement automatique des langues naturelles, plongements de phrases, apprentissage profond, réseaux de neurones structurés.

Title: Sentence embeddings and their relation with sentence structures

Abstract: Historically, models of human language assume that sentences have a symbolic structure and that this structure allows us to compute their meaning by composition. In recent years, deep learning models have successfully processed tasks automatically without relying on an explicit language structure, thus challenging this fundamental assumption. This thesis seeks to clearly identify the role of structure in language modeling by deep learning methods. The dissertation specifically investigates the construction of sentence embeddings—semantic representations based on vectors—by deep neural networks. Firstly, we study the integration of linguistic biases in neural network architectures to constrain their composition sequence based on a traditional tree structure. Secondly, we relax these constraints to analyze the latent structures induced by the neural networks. In both cases, we analyze the compositional properties of the models as well as the semantic properties of the sentence embeddings. This thesis begins with an overview of the main methods used to represent the meaning of sentences, either symbolically or using deep learning. The second part proposes several experiments introducing linguistic biases in neural network architectures to build sentence embeddings. The first chapter explicitly combines numerous sentence structures to build semantic representations. The second chapter jointly learns symbolic structures and vector representations. The third chapter introduces a formal framework for graph transformers. Finally, the fourth chapter studies the impact of the structure on the generalization capacity of the models and compares their compositional capabilities. The last part compares the models to larger-scale approaches. It seeks to discuss current trends favoring larger models, more easily parallelized and trained on more data, at the expense of finer modeling. The two chapters from this part report on the training of large models of automatic language processing and compare these approaches with those developed in the second part from a qualitative and quantitative point of view.

Keywords: Natural language processing, sentence embeddings, deep learning, structured neural networks.

Contents

1 Introduction

> *Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation.*
>
> — **Noam Chomsky**
> For reasons of state, 1973

Linguistic theory is founded on the hypothesis that language has a structure. In computational linguistics, a strong premise is that this structure is recursive (Chomsky 1956) and, in the specific case of sentences, this structure forms a tree. These premises are the cornerstone of linguistic theory. Recently, a new family of methods truly changed the field of computational linguistics by modeling language with vectors. First, word embeddings emerged (Mikolov, K. Chen, et al. 2013; Mikolov, Sutskever, et al. 2013) and, as deep learning gained momentum, it soon became natural to model sentences or even longer texts with vector representations (Cho et al. 2014; Hochreiter and Schmidhuber 1997). In this context, this dissertation is about creating sentence embeddings through the composition of lexical units. Text representation is at the core of natural language processing (NLP), which develops automatic methods for inferring related attributes from those representations. Attributes can take many forms: a given class in a classification problem, the answer to a question, a list of documents with similar semantic content, or a summary of the input text. In recent years, the representation methods for text have developed significantly. Contrary to formal linguistic frameworks, which derive syntactic and semantic properties from expert rules, these methods derive representations by exploiting the implicit patterns within vast corpora. These methods are grounded on two foundational hypotheses: the distributional hypothesis to build word representations given their context and the compositionality principle to combine those words into sentence representations.

This dissertation focuses on sentences and the methods to encode them. Specifically, *(i)* how layouts identified by distributional methods from vast corpora relate to linguistic structures and, respectively, *(ii)* how we can efficiently infuse linguistic biases in neural architectures to drive the composition function learned by self-supervised sentence embedding methods.

This section introduces the study by first discussing the applications of sentence embedding methods, the background, and context, followed by the research problem, the research aims, objectives and questions, the significance, and finally, the limitations.

1.1 Background to the study

Embedding a sentence consists of assigning it to a static, fixed-length, real-valued vector, which captures its meaning. It is important to emphasize the distinction between sentence embedding and any sentence vector representation, for example, intermediate representations from neural networks. The majority of modern NLP methods works end-to-end: the intermediate representations and the inference of attributes from those representations are part of a unified process. In such a case, it is impossible to separate the representations from the final model outputs. The representations, therefore, depend on the attribute we seek to predict and will most likely only capture the information relevant for this prediction. In the context of sentence embedding, the representation of the input text and the inference of related attributes are two explicitly disconnected steps. Therefore, sentence embedding should capture an exhaustive perspective of the text input meaning as we may use them to predict a large variety of attributes. Sentence embedding methods should be highly generic, and their conception should be independent of their later use.

We can divide the properties we expect from sentence embeddings into two categories:

- First, the notion of semantic distance in the original sentence space should be reflected in the representation space. In the sentence embedding space, it is straightforward to define a notion of mathematical

distance over the vector space. Therefore, we can use standard mathematical operators to compare semantic sentence characteristics directly in the sentence embedding space. Sentences with close meanings should be mapped to close embedding vectors.

- Second, we expect sentence embeddings to fully capture the meaning and general characteristics—such as the sentence length or the main verb tense—of the original sentence. It should be possible to extract some specific information from the embedding vector using statistical methods.

Therefore, sentence embeddings are essential for many unsupervised applications such as search engines, information retrieval, text mining, and documents clustering. It is also possible to use sentence embeddings as features for more supervised models, inferring relationships such as entailment between sentence pairs.

1.2 Research problem

Embedding sentences is an active subject of research, including the development of self-supervised training objectives, training datasets, evaluation benchmarks, or the release of models as open-source contributions.

Building standalone sentence embeddings is specifically hard, as an infinite number of valid sentences exist. Compositional semantics state that the meaning of a phrase is determined by combining the meanings of its sub-phrases. Models, therefore, need to compose text units, given a syntactic structure, into global semantic embeddings. However, many contributions rely on standard encoder architectures and do not question the composition mechanisms transforming text units into a global sentence representation.

Thus, sentence embedding methods present pitfalls that are common to many domains in NLP: lack of robustness toward out-of-domain generalization, shallow pattern matching rather than compositional knowledge, the requirement for large training datasets, or over-parametrization.

1.3 Research aims, objectives, and questions

This study aims at improving sentence encoder compositional abilities. We seek to leverage both the integration of linguistic biases into neural network architectures as well as the scaling of these models and their training setup. We define the following research objectives:

1. Develop efficient methods to integrate linguistic biases into neural networks;
2. Evaluate the effectiveness of these strategies and approaches;
3. Compare and contrast these strategies and approaches in terms of their strengths and weaknesses.

The dissertation studies sentence embeddings and their relation with sentence structures. Below, we detail the research questions we investigate. The three first questions focus on the methods to efficiently induce linguistically driven insights within neural network composition functions. The last question asks about the benefits of such insights regarding the generalization power of neural networks for automatic language processing.

Can we efficiently introduce linguistic biases within neural network architectures? The recursive structure of language is a strong hypothesis in computational linguistics (Chomsky 1956). Thus, computing sentence semantic representations traditionally calls for a recursive compositional function whose structure is tree-shaped. In contrast, recent deep learning architectures—such as recurrent neural networks (Cho et al. 2014; Hochreiter and Schmidhuber 1997) or transformers (Vaswani et al. 2017)—encode text without explicit hierarchical composition. The first research question focuses on bridging the gap between these two paradigms: we explore the feasibility of explicitly integrating linguistic priors within neural architectures to compose semantic representations with a hierarchical structure.

Does self-supervised training induce structure into neural architecture? Our second research question has the same

objective—integrating linguistic priors within neural architectures—but different means. This time we are not focusing on the architectures of neural networks but rather the training methods. We explore the possibility of inducing latent structure within the function that neural networks operates to compose lexical units into sentence representations.

Can we induce specific compositional abilities through neural architectures? We explore the possibility of exploiting the model and training dataset size to induce linguistic structure into neural networks. While the size of the language model in natural language processing is steadily increasing (Brown et al. 2020; Devlin et al. 2019), we investigate how such approaches can compensate for the lack of linguistically based insights.

Can we balance the lack of linguistic insights with larger models and larger training datasets? Finally, we investigate the role of structure in building more robust neural network architectures. By influencing the semantic composition of neural networks, we aim to improve their compositional and generalization abilities.

1.4 Significance

We expect this study to contribute to the body of knowledge on sentence embeddings and neural model architectures to encode text. The publications and the ongoing experiments will contribute to the academic effort in building more robust statistical models by incorporating language biases and approaches for scaling model training.

This thesis is funded by Quantmetry, a French pioneering consulting firm working on end-to-end AI projects—from strategy to industrialization.[1] As such, this work is also motivated by the desire of achieving industrial contributions and ready-to-use tools available for real-world applications. Besides empirical research, a large portion of this work is thus released as open-source contributions. Resources include pre-trained language models for English and French, training and evaluation datasets, as well as associated scripts to reproduce the results.[2] Finally, we released a code for

1: https://www.quantmetry.com/

2: https://huggingface.co/asi/gpt-fr-cased-base

recursive models under a library called PyTree.[3] The library was distinguished and listed among the winners of the PyTorch Hackathon 2021. We hope this empirical work and the resources will provide real-world value for organizations in a field in which knowledge and methods are undergoing rapid and continuous evolution.

3: https://github.com/AntoineSimoulin/pytree

1.5 Limitations

Our study is limited to sentences, but we hypothesize that it may, in some cases, extend to paragraphs. However, the major part of our work will not apply to longer chunks of text. Although we seek to propose methods applicable to various languages, the study focuses mainly on English and French, and some experiments may be difficult to reproduce in low-resource languages. Indeed, we make use of specific training and evaluation resources.

Our study proposes efficient methods to introduce linguistic biases into neural models and better characterize model compositional behavior. Such approaches appear promising to avoid unwanted behavior for real-world applications. However, our study also underlines the long road ahead to fully realize the promises of current language models.

1.6 Contributions and Outline

1.6.1 Jointly learning model structure and compositional operations

First, we focus on tree-structured neural networks, which naturally encode the structure of language. For each sentence, the network computes text units following a syntactic tree, starting from the leaf nodes up to the root. However, such models suffer from practical constraints that limit their application. In particular, tree-based models not only require raw text as input but also the sentence structure in the form of a parse tree. Such structure may be tedious because it requires manual annotations and external parsers. To overcome such limitations, we formulated a novel tree-based model that learns its composition function together with its structure.

The model includes two modules, a biaffine graph parser, and a Tree-LSTM. The parsing and the composition functions are explicitly connected and, therefore, learned jointly. The method differs from previous work as the representation is not computed from the whole forest of potential trees. Moreover, training the full model directly does not require supervision from an explicit parsing objective. The model outperforms tree-based models relying on external parsers on downstream tasks. In some configurations, it is even competitive with BERT-base.

1.6.2 Studying shallow structure in transformer models

Recent transformer architectures have gained increased popularity within the community. Their composition function does not require hand-annotated data (like trees) to be trained, unlike tree-based models. On the other hand, as many results suggest, these new models acquire some sort of hierarchical structure. Transformers update each token hidden simultaneously through a fixed number of layers. However, the role of these layers and how they process information is not fully understood. We formulate the hypothesis that the distinct layers do not encode specific surface, syntactic nor semantic functions but rather that such information emerges through the iterative application of layers. To better study the transformation of token representations across layers, we propose a variant of ALBERT (Simoulin and Crabbé 2021b). This model implements the key specificity of weights tying across layers but also dynamically adapts the number of layers applied to each token. We analyze token transformation across the network depth. In particular, we study how iterations are distributed given the token dependency types. We show that tokens do not require the same amount of iterations and that difficult or crucial tokens for the task are subject to more iterations.

1.6.3 Characterizing compositional properties of neural architectures

While transformers show outstanding performance on many NLP benchmarks, they also have some linguistic limitations.

In particular regarding their ability to generalize outside their training range and to learn elementary composition rules. The benchmark COGS (Kim and Linzen 2020) for example, highlights that deep learning models struggle to generalize to longer sequences or sentences with deeper level of recursion than seen during training. Following our work on integrating the structure into neural architecture, we aim at better characterizing how the model structure may affect its degree of compositionality. This work is currently in an experimentation phase. We are building an evaluation setup with arithmetic expressions containing specific properties. We train various models on specific subsets and observe how models generalize outside their domain. Specifically, we compare models with varying structural constraints, such as sequential, recursive, or unstructured models.

1.6.4 Training sentence embedding models using a discriminative objective

Inspired by linguistic insights, we assume structure is crucial to building consistent representations. We indeed expect sentence meaning to be a function of both syntax and semantic aspects. In that regard, we propose a self-supervised method that builds sentence embeddings from the combination of diverse explicit syntactic structures of a sentence. The novelty consists in jointly learning structured models in a contrastive multi-view framework that induces an explicit interaction between models during the training phase. We pre-train various models using a contrastive objective with a 40 million sentences corpus. We then evaluate our model on sentence embedding benchmarks and obtain state-of-the-art results, in particular on tasks that are expected, by hypothesis, to be more sensitive to sentence structure. We relate the development, training, and release of large, state-of-the-art, sentence embedding models. We use a similar contrastive objective and train models on a one billion sentences corpus. We develop specific evaluation benchmarks for sentence embeddings and obtain state-of-the-art results.

1.6.5 Training the first large incremental language model for French

As observed in Linzen and Baroni (2020), deep neural networks have exceptional grammatical competencies. For example, GPT-2 generates correct text with plural and long-distance agreement despite any prior linguistic knowledge. Such agreements are determined by abstract structures and not just the linear order of words. Surprisingly, models can learn such specific linguistic patterns (subject-verb, noun-adverb, verb-verb) with no prior information about linguistic theory. Within our laboratory, we led the project to train the first large language model in French (Simoulin and Crabbé 2021c). We obtained a dedicated computation grant for the public French HPC computer Jean Zay. The model, equivalent to GPT-2 in English, contains more than one billion parameters. We build a dedicated training corpus and parallelize the training between multiple nodes and compute units. We released the model in Open-Source for research and business application purposes.

1.6.6 Outline of the dissertation

We organize the dissertation into three parts.

Part I provides the necessary background in meaning representation, sentence embeddings, and neural model encoders. Chapter 2 introduces meaning representations. Chapter 3 reviews the architecture of standard encoders to compose words into sentence embeddings, the training objective, and evaluation methods.

Part II aims at improving the compositional properties of language models and their ability to generalize outside their training domain. We aim to integrate the recursive property of language within neural models and design architectures based on linguistic theory. Chapter 4 proposes a self-supervised method that builds sentence embeddings from the combination of diverse explicit syntactic structures of a sentence. However, the tree-structured encoders require heavily structured data to compute the semantic representations. In Chapter 5, we propose to overcome this limitation by proposing an architecture inducing trees from raw text and computing semantic representations along with the inferred

structure. Chapter 6 makes the parallel with transformers and sequential or tree-structured models. We interpret transformers as structured neural networks and layers as operations on fully connected graphs. We finally compare all models in Chapter 7 by proposing an in-depth evaluation of their compositional properties.

Part III focuses on training and sharing models at scale. Indeed, the preparation of massive corpora, the training, and the use of large architectures are key for the performance of such models. Chapter 8 presents an attempt to train state-of-the-art sentence embedding models on a very large corpus. Chapter 9 proposes to train the first large generative pre-trained model in French.

1.6.7 Publications

This dissertation contains some contributions that we previously published and presented at conferences.

- Chapter 4 is an extended version of an article published in EACL Student Research Workshop 2021 (Simoulin and Crabbé 2021a). We open-sourced the code developed for recursive models under a library called PyTree.[4] The library was distinguished and listed among the winners of the PyTorch Hackathon 2021;
- Chapter 5 presents work currently under submission;
- Chapter 6 is an extended version of an article published in ACL Research Student Workshop 2021 (Simoulin and Crabbé 2021b);
- Chapter 7 presents original unpublished work as well as work currently under submission;
- Chapter 8 describes the development of state-of-the-art sentence embedding models as part of the project *Train the Best Sentence Embedding Model Ever with 1B Training; Pairs*. This project took place during the *Community week using JAX/Flax for NLP & CV* organized by Hugging Face. Our project was among the competition winners and received an honorable mention;
- Chapter 9 is an extended version of an article published in TALN 2021 (Simoulin and Crabbé 2021c).

4: https://github.com/AntoineSimoulin/pytree

The code used for all experiments carried out for this dissertation, the pre-trained models, the evaluation and training

datasets have been made publicly available as free software through the following repositories:

- https://github.com/AntoineSimoulin/pytree
- https://github.com/AntoineSimoulin/gpt-fr
- https://huggingface.co/datasets/asi/wikitext_fr
- https://huggingface.co/asi/gpt-fr-cased-base
- https://huggingface.co/asi/gpt-fr-cased-small

Background

Defining sentence meaning 2

> *It was six men of Indostan*
> *To learning much inclined,*
> *Who went to see the Elephant*
> *(Though all of them were blind),*
> *That each by observation*
> *Might satisfy his mind.*
>
> *The First approached the Elephant,*
> *And happening to fall*
> *Against his broad and sturdy side,*
> *At once began to bawl:*
> *"God bless me!—but the Elephant*
> *Is very like a wall!"*
>
> *The Second, feeling of the tusk,*
> *Cried: "Ho!—what have we here*
> *So very round and smooth and sharp?*
> *To me 't is mighty clear*
> *This wonder of an Elephant*
> *Is very like a spear!"*
>
> *The Third approached the animal,*
> *And happening to take*
> *The squirming trunk within his hands,*
> *Thus boldly up and spake:*
> *"I see," quoth he, "the Elephant*
> *Is very like a snake!"*
>
> [...]
>
> **— John Godfrey Saxe**
> The Blind Men and the Elephant,
> 1872

Language is perhaps the most distinguishing characteristic between humans and other animals. Any individual of the homo sapiens species can speak or write and understand other species members who know that language. As artificial intelligence rises, the question remains as to whether computer programs can "handle" such a unique ability.

Human beings interact through language: they encode semantic information using words and sentences that others can decode and understand. Amazingly, humans can understand any new sentence and decode the semantic information it carries. However, the latent process remains largely unknown, and it is difficult to express the meaning of a sentence

using a medium other than its sequence of words.

As stated in Chapter 1, this dissertation focuses on sentence embeddings, *i.e.* mapping sentences to vectors, which capture their meaning. This section outlines the general frame and the main hypothesis that underlie our research. We accept these premises as true throughout the remainder of the dissertation and do not discuss them further. In Section 2.1, we first aim to approximate or circumvent the notion of meaning and briefly enumerate the main philosophical trends defining this notion. Regardless of the definition we choose for meaning, Section 2.2 examines the properties of language that semantic representations should verify. Finally, Section 2.3 enumerates representation methods effectively verifying such properties. We distinguish formal representations based on rules from distributed semantic representations.

2.1 Language and meaning

At the beginning of this section, the quote is an Indian parable about blind men who meet an elephant for the first time and touch it, learning and imagining what it is like as they go. Each blind man can feel only one part of the elephant's body, like the side or the tusk. According to their limited experience, they describe the elephant differently. The size of an elephant and the body of work to define the nature of meaning are hardly comparable notions. Nonetheless, this section proceeds like the blind men from India and touches the body of philosophical literature in several parts to introduce several distinctive definitions of linguistic meaning. The purpose of the dissertation is not to argue about these definitions. We will simply assume that one definition holds and that it is indeed possible to define the meaning of a given expression or sentence.

2.1.1 Idea theory

The 17th-century empiricist John Locke is the primary defender of the idea theory of meaning, relating meaning to subjective ideas. In Locke (1847), he defines ideas as mental representations as "whatsoever the mind perceives in itself, or is the immediate object of perception, thought, or

understanding". Simple ideas can be derived into complex ones by composition, comparison, and abstraction. To Locke, "meaning" is the idea one associates with an expression in his mind. Effective communication requires the listener to decode the speaker's words into their associated meanings.

2.1.2 Sense and reference

Direct reference theory Direct reference theory investigates how language interacts with the world. It connects words to the world by defining the meaning of an expression with what it points out in the world. Following the 19th-century philosopher John Stuart Mill, the 20th-century philosopher Bertrand Russell advocates that the meaning of an expression is not what it points out in the mind but instead in the world.

Truth-conditional theory In the 19th century, mathematician and philosopher Gottlob Frege contends with direct reference theory. Frege consents that words refer to something external in the world, but he argues that the meaning of a name extends beyond its referent. In Frege's view, the meaning of an expression or a sentence consists of two elements: a referent and what he called a "sense" (Frege 1892). The sense of an expression is not the thing it refers to but rather how it refers to it. We may determine a single referent by more than one sense, though each sense determines a single referent. "Charles de Gaulle" and the "the first French president elected under the Fifth Republic", for instance, have the same referent but different senses. As such, the sentence "Charles de Gaulle is Charles de Gaulle" results in a tautology, while "Charles de Gaulle is the first French president elected under the Fifth Republic" is truly informative. Within the continuity of Frege's work, the truth-conditional theory of meaning holds sentence or expression meaning to be reducible to their truth conditions. The truth condition is the conditions under which we may evaluate an expression as true or false. The approach is primarily associated with the work of Donald Davidson to apply Alfred Tarski's semantic theory of truth to the semantics of natural language (Davidson 1967).

Inferentialist theory The inferentialist theory holds that meaning results from links between language and experience. A set of "observation sentences" derive their meaning from an account of experiences upon which one can validate such sentences. The meaning of other sentences results from their inferential relations to other expressions.

2.1.3 Usage theory

We commonly associate the use theory of language with the 20th-century philosopher Ludwig Wittgenstein. According to his theory, we do not define words by the mental associations they invoke in our minds nor by the objects they allude to in our world, but by how we use them. Thus, the meaning of the word presupposes our ability to use it. He defines: "For a large class of cases—though not for all—in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language." (Wittgenstein 1953). Wittgenstein argues that the definition of a word arises from the culture and society in which it is used, or as he puts it, from the *"forms of life"*. He emphasizes the intertwinement of language and social situation and, by extension, the social nature of cognition.

2.2 Language properties

Through the dissertation, we will use some notions that we define here.

2.2.1 Entailment and paraphrases

We define *textual entailment* following Dagan et al. (2010) as "Textual entailment recognition is the task of deciding, given two text fragments, whether the meaning of one text is entailed (can be inferred) from another." The entailment notion somehow differs from pure logic inference detailed in Section 2.1.2 and is not bounded to a specific definition of meaning. By extension, we define *paraphrases* as two sentences, A and B, that entail each other.

2.2.2 Compositionality principle

Natural language understanding requires to know the meaning of individual words. For example, English speakers share common concepts associated with "boy", and they know it relates to something different than "house" or "car". Speakers or writers can also combine words into a theoretically unlimited number of valid sentences. This creative aspect of language allows the formation of arbitrary long sentences.[1]

Through the entire dissertation, we will also adhere to the compositionality principle. According to this principle, the meaning of a linguistic expression may be recursively composed of the meaning of its parts. This principle, exposed below, is known as the compositionality principle and is sometimes called "Frege's Principle".[2]

The principle of semantic compositionality:

The meaning of a complex expression is determined by its structure and the meanings of its constituents.

It is important to note that a particular conception of meaning in no way binds the principle of compositionality. It is often reported as a principle that applies to any semantic theory.

The main argument favoring compositionality is the many semantic theories successfully built by linguists upon its basis. Moreover, productivity and systematicity, which constitute weaker versions of the principle, are accepted by a large community. Productivity is defined by Frege as: "the possibility of our understanding sentences which we have never heard before rests evidently on this, that we can construct the sense of a sentence out of parts that correspond to words." (Frege 1914). Systematicity refers to the existence of definite and predictable patterns within the sentences we comprehend. Or, as defined by Cummins (1996) "whenever it can process a sentence s, it can process systematic variants of s, where systematic variation is understood in terms of permuting constituents or (more strongly) substituting constituents of the same grammatical category".

1: For example, the book Zone (Enard 2008) is a lengthy 500 pages monologue. Proust is also well known for its long sentences, 43 words on average. The longest sentence from *À la recherche du temps perdu* (Proust 1921) contains 856 words. In *les Miséarables* (Hugo 1862), Victor Hugo wrote a sentence with 823 words.

2: We extracted this formulation from the Stanford Encyclopedia of Philosophy: https://plato.stanford.edu/entries/compositionality/#ArguForComp.

2.3 Formal semantic representations

Computational linguistics developed formal structures aiming at capturing sentence meaning. *Meaning representation languages* define these structures utilizing syntactic and semantic frameworks. These frameworks may extend beyond the representation of individual sentences to incorporate common-sense knowledge of some world. Moreover, such representations have convenient properties and may be used for complex natural language understanding tasks such as question answering (Pasupat and P. Liang 2015), robot navigation (Artzi and Zettlemoyer 2013) or database querying (Zelle and Mooney 1996).

Assigning representations to linguistic inputs is known as semantic parsing or semantic analysis. Many representation frameworks exist for representing a text's meaning. This section briefly describes standard meaning representations and outlines their main characteristics, limitations, and applications. We review logic-based, graph, and programming languages formalisms.[3]

3: More details may be found here: https://web.stanford.edu/~jurafsky/slp3/15.pdf.

2.3.1 Logic-based formalisms

First order logic (FOL) represents an instance of a specific object in the world being described using constants. As in the truth-conditional theory detailed in Section 2.1.2, FOL representations allow to evaluate a sentence as true or false. It can also represent an unspecified object of a given type using variables. Finally, it can describe the relationship between objects using predicates. FOL is often described as a good compromise between expressiveness and tractability. For example, the simple sentence "a man is eating a tomato" can be mapped to FOL:

$$\exists x(\exists y(man(x) \land eat(x,y) \land tomato(y)) \tag{2.1}$$

2.3.2 Graph-based formalisms

First-order logic is a very generic system used in mathematics, philosophy, linguistics, and computer science. The represen-

tations we enumerate below are specific to computational linguistics.

Abstract Meaning Representations (AMR) represent sentence semantics using rooted, labeled, directed, acyclic graphs (DAGs). In this representation, nodes correspond to variables referring to entities, events, properties, and states. Edges correspond to relations between the entities. There are around 100 possible relations, including general semantic relations (for example, direction, cause), quantities (for example, unit), dates or lists. The graph does not necessarily follow the syntactic sentence structure and two sentences with similar meaning, but different wording, may be mapped to the same AMR.

AMR have gained in popularity thanks to the tools, resources, and documentation formatted in Banarescu et al. (2013). Today tools include libraries for parsing, visualization, surface generation, and publicly available datasets. Many of these resources are collected at the AMR homepage.[4] It is important to note that these resources and the framework formalism are highly biased toward English and are not intended to bridge the gap with other languages.

4: https://amr.isi.edu/

We illustrate AMR by mapping the sentence "The boy wants to go.". This example is extracted from Banarescu et al. (2013) with the following AMR structure. We also illustrate the structure using an equivalent graph representation in Figure 2.1.

```
(w/ want-01
:ARG0(b/boy)
:ARG1(g/go-01
  :ARG0 b))
```

Universal Conceptual Cognitive Annotation (UCCA) is a semantic representation based on directed acyclic graphs (Abend and Rappoport 2013). Terminal nodes can be arbitrary morphemes, words, or multi-word chunks. Inner nodes consist of a single entity defined by semantic or cognitive factors over the connected units. Edge labels represent a child's contribution to the semantics of the parent unit. The

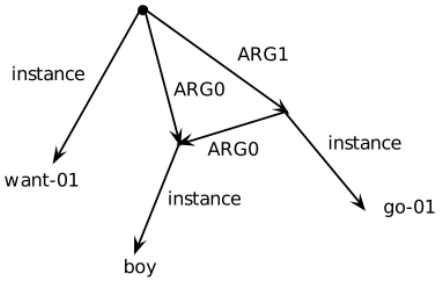


Figure 2.1: Graph visualization of the AMR structure associated with the sentence "The boy wants to go.". Figure extracted from Banarescu et al. (2013).

annotated text is mostly based on English Wikipedia articles with 148 annotated passages (an average length of 385 tokens).

Discourse Representation Structures (DRS) are formal meaning representations developed as part of the Discourse Representation Theory (DRT) (Kamp and Reyle 1993). It aims at better-representing phenomena such as interactions between indefinite noun phrases and (anaphoric) pronouns, treatment of negation, modals, and quantification scope. The specificity of DRT is to propose an interpretation of discourses spanning over more than one sentence and not only individual sentences. It proposes a view of language where its interaction with its context defines the semantics of a sentence. The framework presents similarities with logical formalisms as it enables the evaluation of a sentence's truth value and performs semantic inference. Regarding the representation structure, DRS map the discourse to a graph where nodes are discourse referents representing entities under discussion and edges representing information exchanged between referents.

Semantic Dependency Parsing (SDP) is a formal meaning representation in the form of a directed graph with arcs between pairs of words (Oepen et al. 2015). The vertices between words describe predicate-argument relationships.

2.3.3 Programming languages

Finally, as mentioned in the survey conducted in Kamath and R. Das (2019), a line of work aims at translating natural language into executable functions from general-purpose

programming languages. Such languages have an explicit syntax and are less subject to ambiguity. Many benchmarks exist, in particular, to convert questions into executable SQL queries (Dahl et al. 1994; Finegan-Dollak et al. 2018).

2.4 Practical limits of symbolic representations

The methods presented above represent meaning in the form of dedicated structures. Such frameworks have explicit properties which facilitate their use. For example, SQL queries can be executed on a knowledge base to answer factual questions. However, meaning representations are usually supposed to label a training dataset to learn the mapping of the sentences. Labeling such a dataset is usually complex and requires linguistic experts. The process should be repeated for every domain and language and is usually restricted to generic English data. Finally, the semantic parser is not guaranteed to produce the proper structure and may be subject to errors.

On the other hand, distributional semantic representations propose to represent semantic meaning in the form of a real-value fixed-length vector (Jurafsky and J. H. Martin 2022).[5] As for formal representations, many methods exist to learn this mapping. However, the main idea is to use self-supervised data that do not require heavy labeling effort. In this perspective, we learn to map a sentence to a semantic vector using only the structure and patterns within raw text. In particular, we train such methods to verify specific sub-properties induced by natural language understanding, such as predicting inference or reconstructing a sentence surface form given its embedding. In this work, we will focus on such approaches, and we will provide an in-depth review of such methods in Chapter 3.

5: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Embedding sentences | 3

> *In ancient times, before the advent of mass-market publishing, manuals were written on scrolls. The people believed that kotodama—the soul or spirit of language—resided in every word; that in uttering a thought one gives life to it; that words hold a spiritual power. This belief gave the written and spoken language a near-mystical status and encouraged a reverence for the written word beyond that in the West.*
>
> **— Jake Adelstein**
> Tokyo Vice, 2009

This chapter proposes a literature review on today's state-of-the-art methods to train and evaluate sentence embedding models. We first expose traditional word embeddings methods (Section 3.1). We then enumerate in Section 3.2 methods to compose words into sentence representation. We review the main training and evaluation setups in Section 3.3 and Section 3.4.

3.1 Embedding words

Embeddings are today the cornerstone of every neural language model. In mathematics, an embedding is an injective and structure-preserving map e from one mathematical structure X to another Y. The notion of "structure-preserving" depends on the nature of the latter structures. In natural language processing, we define words as a string (a sequence of characters) and the vocabulary as a finite set of distinct words. Embeddings e map the vocabulary V to a vector space E of dimension h. e is an injective function, and therefore, each word w from the vocabulary is mapped to precisely one unique vector. All vectors from E have a fixed length h and real values and are thus sometimes called continuous vectors.

Embeddings are convenient as we can exploit all the built-in properties from the representation space E. It thus provides

all the mathematical tools to analyze words without relying on their surface form. It is straightforward to define a notion of distance over the representation vector space to characterize its geometry. It is less obvious on the original vocabulary space. Embedding methods usually rely on the distributional hypothesis: they characterize words given their distributions of co-occurrences in a given corpus. The core idea is that words with similar meanings tend to appear in similar contexts. As mentioned, embeddings preserve the structure from the original space. Therefore, embedding methods ensure that words with close distributions are mapped to close vectors, while words with distant distributions are mapped to distant vectors.

There exist multiple embedding frameworks. Since the 1990s, vector space models have become a popular tool in distributional semantic analysis, particularly with Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). Collobert and Weston (2008) introduced a neural network architecture that formed the basis for many current methods utilizing pre-trained word embeddings. Their widespread application was enabled by *word2vec* (Mikolov, K. Chen, et al. 2013; Mikolov, Sutskever, et al. 2013) and GloVe (Pennington, Socher, and Manning 2014), efficient frameworks for the training of pre-trained embeddings. Word embeddings are characterized by their self-supervision. They only need raw corpora of text to be trained. It is also possible to use layers that learn embeddings together with a given downstream task without prior training.

3.2 Composing words into sentence embeddings

Many modern NLP systems use word embeddings as base features. Generalizing to embeddings for larger chunks of text, such as sentences, remains a question to be solved. Word embeddings operate on a finite vocabulary set, while we may build an infinite number of valid sentences. We can, therefore, not directly extend methods for embedding words into sentences. Sentence embedding methods aim to exploit the compositionality principle: they compose word vector representations into semantic sentence representations.

Artificial neural networks consist of connected units called neurons. Neurons define a vector space transformation based on linear algebra operators and nonlinear activation functions. Neural networks typically contain a very large number of neurons, which may be arranged into layers. Neurons—and by extension, layers—are interconnected: they receive input from their inner connections and send their output to their outer connections. Each layer has its own inner structure and connection pattern. This section presents standard NLP architectures and defines the notations we will reuse in all further chapters.

3.2.1 Bag-of-Words

The most straightforward method to combine word vectors is the Bag-of-Words (BoW). We simply average all the vectors from the sentence into one vector of the same size. This method does not account for the order of the words in the sentence nor any kind of sentence structure. However, as analyzed in Arora, Y. Liang, and Ma (2017), this simple method is a strong baseline for producing sentence embeddings.

3.2.2 Recurrent neural networks

Recurrent neural networks (RNN) (Cho et al. 2014; Hochreiter and Schmidhuber 1997) take sequences $X = (x_1, x_2 \cdots x_T)$ as input. As illustrated in Figure 3.1, they process the sequence iteratively, starting from the first element of the sequence to the last. The network consists of a RNN cell. For each element of the sequence x_t, the cell outputs an hidden state h_t, which depends from the current element of the sequence x_t and from the previous element hidden state, h_{t-1}. The cell parameters are shared between each step, and RNN can process sequences of arbitrary length.

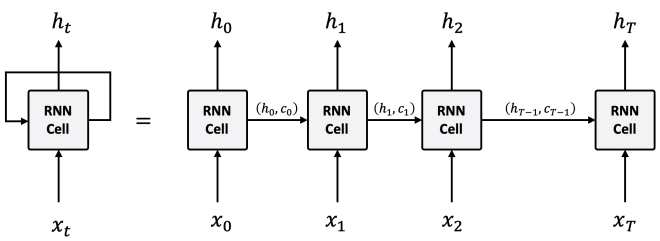


Figure 3.1: We illustrate the recursive application of the RNN cell.

Basic recurrent neural networks suffer from practical limitations. In particular, gradient over-flow or underflow: when propagating the gradient error through the sequence, it tends to become very small or very large. Gated mechanisms can mitigate this problem, as these gates determine which information to retain for each time step.

Gated recurrent units (GRU) include a reset and update gate. Intuitively, the reset gate r determines which information from previous step to reset (Equation 3.3). The update gate z, determines the amount of previous information that passes along the next step (Equation 3.4).

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)}\right), \tag{3.1}$$

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}\right), \tag{3.2}$$

$$\tilde{h}_t = \tanh(W^{(h)}x_t + U^{(h)}(r_t \odot h_{t-1}) + b^{(h)} \tag{3.3}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{3.4}$$

Long short-term memory (LSTM) integrates three gates. Besides the short memory vector h, it adds a long-term memory vector c that is passed along the steps. We detail the memory mechanism in Equation 3.5 to Equation 3.10. Intuitively, the input gate i determines what information to store in long-term memory. The forget gate f determines which information from the long-term memory to forget. Finally, the output gate o computes the new short-term memory to balance the current input, the previous short-term memory, and the newly computed long-term memory.

$$i_t = \sigma\left(W((i)x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \tag{3.5}$$

$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right), \tag{3.6}$$

$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right), \tag{3.7}$$

$$u_t = \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right), \tag{3.8}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1}, \tag{3.9}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.10}$$

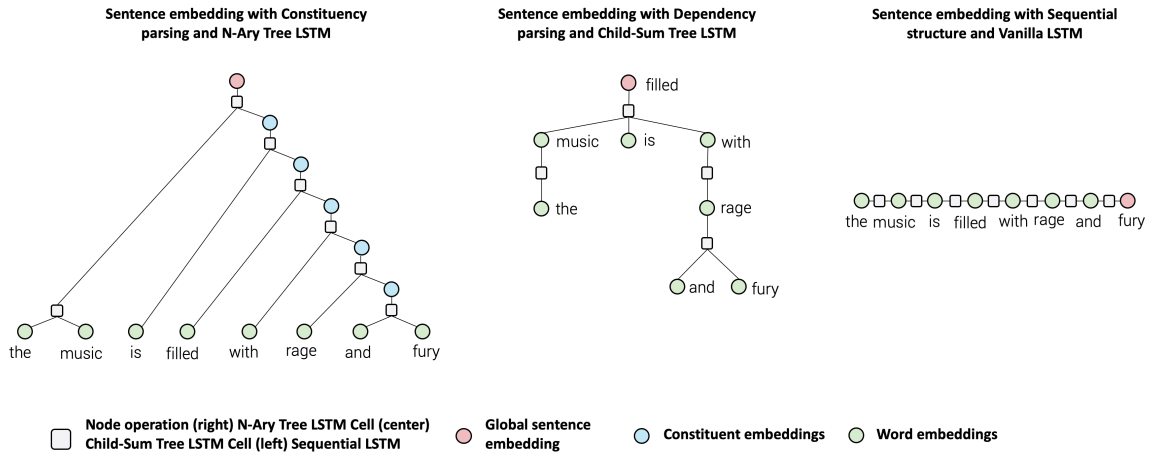


Figure 3.2: *(left)* The sentence is parsed in constituency and the tree is binarized. The application of the N-Ary Tree LSTM on the obtained structure is represented. *(center)* The sentence is parsed in dependency and a Child-Sum Tree LSTM model is recursively applied. This example illustrates the structural difference between these two views. Dependency parsing is articulated around the verb "filled", which is the root node. In constituency, subject and verb are connected through the root node. The two architectures differ as the N-Ary Tree LSTM is structured as a binary tree and differentiates the left and right children, while the Child-Sum Tree LSTM might have an arbitrary number of unordered nodes. *(right)* The sentence structure follows the linear order of the words and is encoded using a standard sequential recurrent network.

3.2.3 Tree-structured neural networks

Tree-structured neural networks generalize sequential networks to tree-structured topologies. We illustrate the comparison between various structures in Figure 3.2. They also consist in a cell that composes a state from an input vector x_j and the hidden states of the input children, $h_k, \forall k \in C(j)$ with $C(j)$ the children of node j. As such, a sequential RNN is a special case of a Tree-RNN, where every node has exactly one child. We illustrate the composition process along with an arbitrary tree structure in Figure 3.3.

From a practical point of view, implementing tree-structured models can be challenging. We open-sourced the code we developed for recursive models under a library called PyTree.[1] The library was distinguished and listed among the winners of the PyTorch Hackathon 2021.[2]

1: https://github.com/AntoineSimoulin/pytree

2: https://pytorch2021.devpost.com/

Intuitively, tree-structured networks may be a better fit for language, which is supposed to follow a recursive structure. Figure 3.4 provides examples of the stanford sentiment treebank, and highlights the importance of considering the sentence structure to predict the sentiment from a complete sentence.[3] Indeed, isolated words may be negative, while the

3: https://nlp.stanford.edu/sentiment/index.html

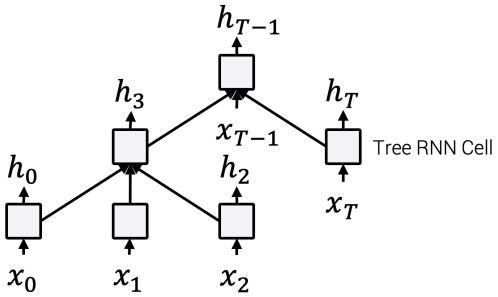


Figure 3.3: We illustrate the application of the Tree LSTM on an arbitrary branching tree. The figure takes inspiration from Tai, Socher, and Manning (2015).

entire sentence will be positive.

We focus on two specific frameworks describing language structure: dependency and constituency parsing. In constituent analysis, the syntactic structure of a sentence is represented as nested multi-word constituents. The dependency tree represents the relationship between individual words. For constituent analysis, it is possible to binarize the tree, such that every node has exactly two children. It is also possible to differentiate the left and right children. Given this distinction, we define two tree-structured cell operations adapted for each framework.

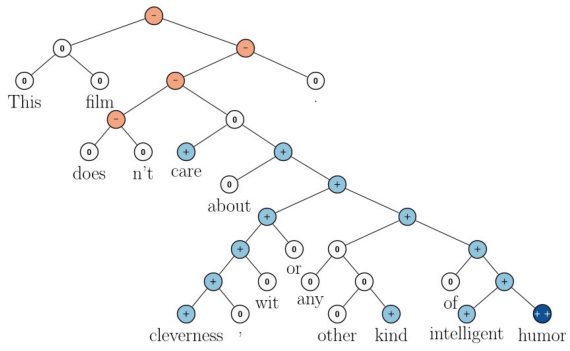


Figure 3.4: We illustrate the benefits of tree structure encoding. The negation score impacts the full sentence sentiment prediction. We extract the figure from Socher, Perelygin, et al. (2013).

Childsum Tree LSTM Tai, Socher, and Manning (2015) compute sentence embeddings using a recursive node function derived from standard LSTM formulations but adapted for tree inputs. Each node is assigned an embedding given its dependents with a recursive function. The hidden state is computed as the sum of all children's hidden states (Equation 3.11). This model is adapted for dependency tree structures in which words are connected through dependency edges. A word may have an arbitrary number of dependents.

$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \tag{3.11}$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \tag{3.12}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \tag{3.13}$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \tag{3.14}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \tag{3.15}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{3.16}$$

$$h_j = o_j \odot \tanh(c_j) \tag{3.17}$$

With $C(j)$, the set of children of node j. In Simoulin and Crabbé (2021a), we propose an Attentive Child-Sum Tree LSTM and we compute $\tilde{h}_j$ as the weighted sum of children vectors as in Y. Zhou, C. Liu, and Pan (2016). We replace computation of $\tilde{h}_j$ in Equation 3.11 with Equation 3.18 that allows the model to filter semantically less relevant children.

$$\tilde{h}_j = \sum_{k \in C(j)} \alpha_{kj} h_k \tag{3.18}$$

The parameters α_{kj} are attention weights computed using a *soft attention layer*. Given a node j, we consider $h_1, h_2, \ldots, h_n$ the corresponding children's hidden states. the soft attention layer produces a weight α_k for each child's hidden state. We did not use any external query to compute the attention but instead use a projection from the current node embedding. The attention equations are detailed below:

$$q_j = W^{(q)} x_j + b^{(q)}; \quad p_k = W^{(p)} h_k + b^{(p)} \tag{3.19}$$

$$a_{kj} = \frac{q_j \cdot p_k^\top}{\|q_j\|_2 \cdot \|p_k\|_2} \tag{3.20}$$

$$\alpha_{kj} = \text{softmax}_k(a_{1j} \cdots a_{nj}) \tag{3.21}$$

N-ary Tree LSTM is also defined in Tai, Socher, and Manning (2015). It is a tree-structured model designed for constituency parsed inputs, which describes the sentence as a nested multi-word structure. In this framework, words are

grouped recursively in constituents. Only leaf nodes correspond to words in the resulting tree, while internal nodes encode word sequences recursively. It is possible to binarize the trees to ensure that every node has exactly two dependents. Again the representation is computed bottom-up, and the embedding of the tree root node is used as sentence embedding. The equations make the distinction between right and left nodes.

$$i_j = \sigma\left(W^{(i)}x_j + \sum_{\ell=1}^{N} U_\ell^{(i)} h_{j\ell} + b^{(i)}\right), \tag{3.22}$$

$$o_j = \sigma\left(W^{(o)}x_j + \sum_{\ell=1}^{N} U_\ell^{(o)} h_{j\ell} + b^{(o)}\right), \tag{3.23}$$

$$u_j = \tanh\left(W^{(u)}x_j + \sum_{\ell=1}^{N} U_\ell^{(u)} h_{j\ell} + b^{(u)}\right), \tag{3.24}$$

$$f_{jk} = \sigma\left(W^{(f)}x_j + \sum_{\ell=1}^{N} U_{k\ell}^{(f)} h_{j\ell} + b^{(f)}\right), \tag{3.25}$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell}, \tag{3.26}$$

$$h_j = o_j \odot \tanh(c_j), \tag{3.27}$$

3.2.4 Transformer neural networks

Introduced in Vaswani et al. (2017), transformers originally consisted of an encoder-decoder framework relying almost exclusively on attention and completely discarding any recurrent operation. By extension, the encoder or decoder taken separately may also be called transformers, and we focus here on the encoder part. Transformer implementations may easily be parallelized since layers compose token contextualized representations simultaneously. We illustrate the architecture in Figure 3.6 with a focus on the inner layer architecture in Figure 3.5.

As usual, the first layer (Equation 3.28) is an encoding layer that maps each word from a sequence $\{u_1 \cdots u_T\}$ to a corresponding embedding. Additionally, the embedding layer encodes each word position with dedicated positional embedding weights.

$$h_t^0 = W^{(e)} u_t + W^{(p)}, \tag{3.28}$$

With $W^{(e)}$ the embedding matrix, and $W^{(p)}$ the positional embedding matrix.

Transformers are composed of a series of layers. Each layer acts as a many-to-many encoder, mapping a set of vectors $\{h_t^n\}_{t \in [\![1,T]\!]}$ to a set of so-called contextualized vectors $\{h_t^{n+1}\}_{t \in [\![1,T]\!]}$. Each layer is composed of a multi-head attention layer (Equation 3.29 to Equation 3.32) that maps each input vector to a weighted average from the input set, followed by a feed-forward network (Equation 3.33 to Equation 3.35).

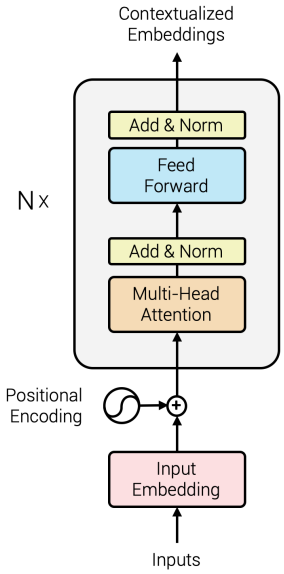


Figure 3.5: Illustration of encoder part of the transformer inner layer architecture.

$$H = [h_t, \quad \forall t \in [\![1, T]\!]] \tag{3.29}$$

$$Q_h = HW_Q^{(h)}, \quad K_h = HW_K^{(h)}, \quad V_h = HW_V^{(h)}, \tag{3.30}$$

$$\alpha_{i,j}^{(h)} = \text{softmax}_j \left(\frac{Q_h K_h^\top}{\sqrt{k}} \right) \tag{3.31}$$

$$h'_t = \sum_{h=1}^{H} W_C^{(h)} \sum_{j=1}^{\top} \alpha_{t,j}^{(h)} V_h \tag{3.32}$$

$$h_t = \text{LayerNorm}(h_t + h'_t; \gamma_1, \beta_1) \tag{3.33}$$

$$h'_t = W_2^\top \, \text{ReLU}(W_1^\top h_t) \tag{3.34}$$

$$h_t = \text{LayerNorm}(h_t + h'_t; \gamma_2, \beta_2) \tag{3.35}$$

With $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d \times k}$, $W_C^{(h)} \in \mathbb{R}^{k \times d}$, $W_1 \in \mathbb{R}^{d \times m}$, $W_2 \in \mathbb{R}^{m \times d}$, and $\gamma_1, \beta_1, \gamma_2, \beta_2 \in \mathbb{R}^d$. H denotes the number of attention heads, d the dimension of the model. We set $k = \frac{d}{H}$. The notation softmax_j indicates we take the softmax (defined in Equation 3.31) over the d-dimensional vector indexed by j. In Equation 3.29, [ ] indicates the vector concatenation operation.

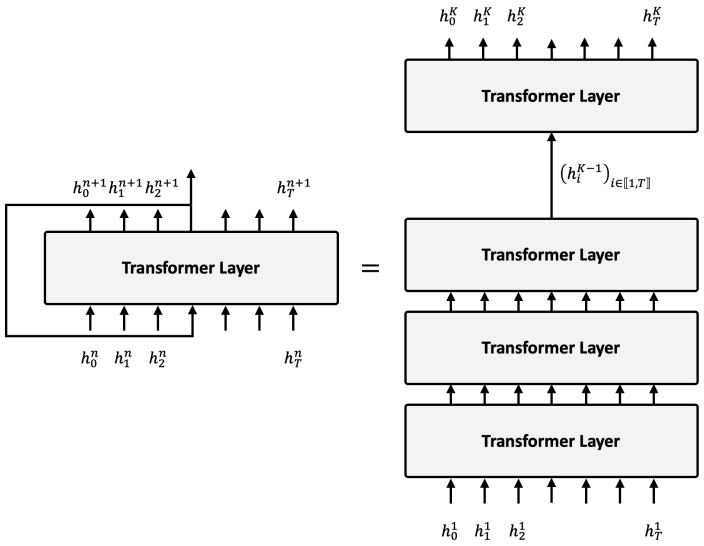


Figure 3.6: We illustrate the iterative application of transformer layers. Contrary to RNN, the weights are usually not shared between layers.

3.3 Training sentence embeddings

Each of the neural network architectures discussed in Section 3.2 takes a sentence as input and composes its inner lexical units into a vector. We can interpret them as a parameterized function f_θ, mapping a sentence s to a vector $h \in \mathbb{R}^d$ of dimension d. However, how can we learn the parameters θ of the function f? In other words, how can we learn a composition function that maps sentences to generic, general-purpose vector representations?

A straightforward training setup would be to use the sentence vector h as input for a *supervised task*. After backpropagating through the entire architecture, we can update the encoder's composition weights. However, such a supervised process is insufficient to produce meaningful sentence embeddings. Indeed, as stated in Section 1.1, sentence embeddings intend to provide generic, general-purpose sentence representations. Since we are training the models with a supervised

objective, the intermediate sentence representations from the encoder are likely to capture properties related to the task. On the contrary, we seek to obtain representations that can be successfully applied to many different tasks.

We could imagine a supervised task that takes a sentence as input and directly outputs its meaning. We could update the model weights and learn a highly generic encoding function by comparing the predicted sentence meaning with some references. However, as discussed in Section 2.1, the notion of meaning is somehow ineffable or ethereal. As an alternative to predicting the sentence's meaning directly, we may consider predicting a related attribute that is well conceivable and expressible. Such an attribute can be a proxy that requires capturing the sentence meaning to be predicted.

Certainly, the nature of the proxy task is crucial to the procedure. This section makes a literature review of the common tasks used as a proxy objective to train sentence embeddings. Most of these proxy tasks involve predicting a relationship between two or more sentences. In theory, the model cannot predict the relationship without fully capturing the meaning of the considered sentences. All proxy objectives may impact the model's capacity to capture aspects of meaning or the amount of data necessary to train a model. In Section 3.3.1, we obtain the relation between the sentences by labeling data. It is also possible to use a weaker signal in a self-supervised setting (Section 3.3.2). It is possible to mix multiple training paradigms in a multi-task setup (Section 3.3.3). Finally, we review a line of work focusing on learning cross-lingual sentence embeddings (Section 3.3.3).

3.3.1 Supervised learning

Infersent In keeping with the definition of meaning discussed in the last chapter, a model that captures the meaning of a sentence could infer the entailment relation between sentence pairs. Thus, training a model to predict the entailment relationship between two sentences seems reasonable to build efficient sentence embeddings. Therefore, in this setup, the proxy task is a natural language inference task (NLI). NLI consists of a supervised classification task. The model takes as input a sentence pair: a premise and an hypothesis. It should then predict whether the first entails, contradicts,

Table 3.1: SNLI examples presented in the original paper (Bowman, Angeli, et al. 2015) and extracted from the development section of the corpus.

Premise	Hypothesis	label
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping	contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	contradiction
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	neutral

or is neutral to the second. Large datasets exist for English like Stanford Natural Language Inference (SNLI) (Bowman, Angeli, et al. 2015)[4] and MultiNLI (Williams, Nangia, and Bowman 2018)[5] or other languages, including French with the XNLI corpus (Conneau, Rinott, et al. 2018). We present some examples from the SNLI task in Table 3.1.

4: The dataset includes 570k pairs of sentences, distributed in a 550k/10k/10k train/dev/test split.

5: The MultiNLI includes 433k sentence pairs. We refer to the concatenation of the SNLI and MultiNLI as AllNLI.

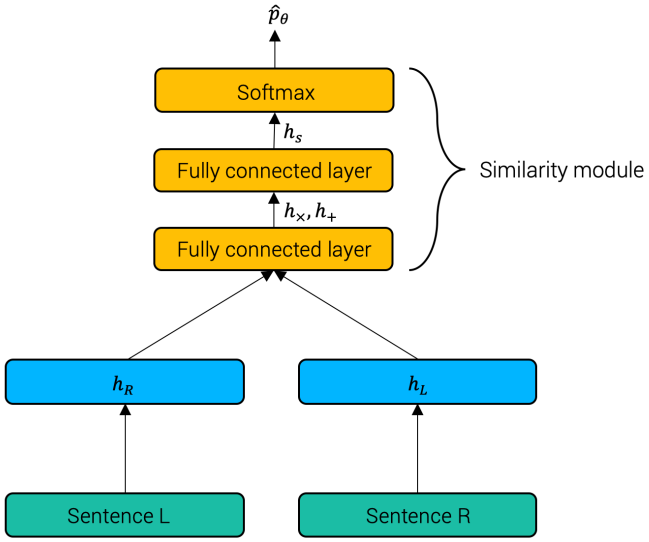


Figure 3.7: Similarity architecture to train models on the SNLI. The encoder networks have tied weights (siamese network structure). The figure is for illustrative purposes only as multiple variations of the similarity module exist.

Conneau, Kiela, et al. (2017) propose a siamese framework to train models on NLI data, illustrated in Figure 3.7. First, a sentence encoder separately encodes the premise h_L and the hypothesis h_R. The encoder weights are shared for the encoding of both parts, but the two sentences are not encoded jointly (as is the case when using cross-features or attention architectures). Then, a dedicated architecture is used to predict the similarity distribution from the pair of sentences. The similarity module s takes as input a pair of sentence vectors h_L and h_R and outputs a vector h_s comparing them

such that $h_s = s(h_L, h_R)$. In Equation 3.36, we report the version of the similarity architecture proposed in Conneau, Kiela, et al. (2017).[6] The module takes as input the embeddings h_L and h_R and computes their componentwise product $h_L \odot h_R$ and their absolute difference $|h_L - h_R|$. Given these features, the module computes the probability distribution $\hat{p}_\theta$ using a three-layer perceptron network (MLP) followed by a softmax:

$$
\begin{aligned}
h_\times &= h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\
h_s &= W^{(1)}[h_\times, h_+, h_L, h_R] + b^{(1)}, \\
h_s &= W^{(2)} h_s + b^{(2)}, \\
\hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}),
\end{aligned}
\tag{3.36}
$$

6: There exists multiple variations of the similarity module which differs given the aggregation function of h_L and h_R, the number of fully-connected layers and their hidden dimensions (Choi, Yoo, and S. Lee 2018; Conneau, Kiela, et al. 2017; Reimers and Gurevych 2019).

Conneau, Kiela, et al. (2017) propose multiple sentence encoders to build h_L and h_R, including LSTM and GRU, BiLSTM with mean/max pooling, Self-attentive network or Hierarchical ConvNet. SentenceBert later adapted the setup to use Bert as sentence encoder (Reimers and Gurevych 2019). Reimers and Gurevych (2019) use the same supervised training method but with a pre-trained Bert as encoder.

Table 3.2: Example pairs presented in DisSent original paper (A. Nie, Bennett, and Goodman 2019). Each pair consist of two sentences linked with discourse relations. The training pairs are collected using a semi-automated procedure.

S1	S2	Marker
Her eyes flew up to his face.	Suddenly she realized why he looked so different.	and
The concept is simple.	The execution will be incredibly dangerous.	but
You used to feel pride.	You defended innocent people.	because
Ill tell you about it.	You give me your number.	if
Belter was still hard at work.	Drade and barney strolled in.	when
We plugged bulky headsets into the dashboard.	We could hear each other when we spoke into the microphones.	so
It was mere minutes or hours.	He finally fell into unconsciousness.	before
And then the cloudy darkness lifted.	The lifeboat did not slow down.	though

DisSent A. Nie, Bennett, and Goodman (2019) propose a weaker signal to train sentence embeddings: the discourse relations between sentences. The task is positioned as an

intermediary between a fully supervised and self-supervised approach. Given two sentence embeddings, a classifier aims to identify which discourse marker was used to link the sentences. As with infersent, the setup can accommodate any sentence encoder, such as sequential LSTMs or larger pre-trained models, such as BERT. We present some examples of the training data in Table 3.2.

7: This model requires a training corpus of contiguous text. The BookCorpus dataset is a collection of 11,038 free books written by yet unpublished authors. It contains books in 16 different genres. It contains 74,004,228 sentences and 984,846,357 words.

The training dataset is built upon the BookCorpus dataset (Zhu et al. 2015).[7] The training pairs are collected using a semi-automated procedure. The authors used the Stanford CoreNLP dependency parser (Schuster and Manning 2016) to identify discourse markers between two sentences S_1 and S_2. They collected a curated dataset of 4,706,292 pairs of sentences for 15 discourse markers. The training procedure is close to infersent. Given a sentence pair, a sentence encoder model produces sentence embeddings (s_1, s_2). A similarity module s computes a similarity vector $h_s = s(s_1, s_2)$. We detail the similarity module used in A. Nie, Bennett, and Goodman (2019) in Equation 3.37. The module computes pairwise vector operations and outputs a probability distribution over discourse relations.

$$
\begin{aligned}
&s_{\text{avg}} = \frac{1}{2}(s_1 + s_2), \quad s_{\text{sub}} = s_1 - s_2, \quad s_{\text{mul}} = s_1 * s_2 \\
&S = [s_1, s_2, s_{\text{avg}}, s_{\text{sub}}, s_{\text{mul}}] \\
&h_s = \text{ReLU}(W^{(2)} h_s + b^{(2)}), \\
&\hat{p}_\theta = \text{softmax}(W^{(p)} h_s + b^{(p)}),
\end{aligned}
\tag{3.37}
$$

Mining sentence pairs It is also possible to use other sentence pairs as signal to train sentence embedding models. To only cite a few, Wieting and Gimpel (2018) produce the PARANMT-50M, a dataset of more than 50 million English-English sentential paraphrase pairs. The dataset was generated automatically using neural machine translation on a parallel corpus. Yang, Cer, et al. (2020) train a multilingual sentence embedding model by using training QA pairs mined from online forums and QA websites, including Reddit, StackOverflow, and YahooAnswers.

3.3.2 Self-supervised learning

Previous methods rely on annotated data or semi-automatically constructed corpora. However, such resources may be hard to find in languages other than English or specific domains. In this section, we review methods that rely only on the structure of raw text, which may be trained in a self-supervised manner.

ParagraphVector (*doc2vec*) Q. V. Le and Mikolov (2014) extend the *word2vec* model (Mikolov, K. Chen, et al. 2013) to learn document-level embeddings. A document is typically understood in the broadest possible sense, encompassing a word n-gram, a sentence, a paragraph, or a large document. The method adds a hierarchical level to *word2vec*, by segmenting the training corpus into paragraphs, themselves tokenized into a sequence of words. The model learns a word matrix W, mapping each word from the vocabulary to a vector w and a paragraph matrix D, mapping each paragraph from the training corpus to a vector d.

This Paragraph Vector model comes in two versions, the Distributed Memory Model (PV-DM) and the Distributed Bag of Words (PV-DBOW), extending respectively the CBOW and Skip-Gram models. As in the original version of *word2vec*, each model consists of a log-linear model trained to predict surrounding words. *doc2vec* adds an additional paragraph token as an input, which acts as a memory to store the context and topic for each paragraph. The paragraph vector matrix is shared between all contexts within a paragraph, but not between paragraphs, whereas the word vector matrix W is shared across paragraphs. In practice, the PV-DM optimizes the loss function from Equation 3.38, that is, the log probability of for each word w_t of each paragraph, knowing the surrounding words $w_{t-k} \cdots w_{t+k}$ and the paragraph d_c. The PV-DBOW optimizes the loss function from Equation 3.39, that is, the log probability of for each word w_t of each paragraph, knowing the paragraph d_c.

$$\mathcal{L}_{\text{PV-DM}} = \frac{1}{D}\sum_{c=1}^{D}\sum_{t=k}^{T_c-k}\log P(w_t|w_{t-k}\cdots w_{t+k}, d_c) \quad (3.38)$$

$$\mathcal{L}_{\text{PV-DBOW}} = \frac{1}{D}\sum_{c=1}^{D}\sum_{t=k}^{T_c-k}\sum_{-k\leq j\leq k, j\neq 0}\log P(w_{t+k}|d_c) \quad (3.39)$$

With D, the number of paragraphs in the training corpus, T_c the number of words for the paragraph at index c, d_c the embedding vector from the paragraph at index c.

The method is conceptually straightforward but has practical limitations. To determine the paragraph vector for a new paragraph, we must perform an inference step that only updates the paragraph matrix D; the word vectors W and the parameters for the rest of the model are fixed.

Skip-thought (ST) R. Kiros et al. (2015) aim at translating the skip-gram function to the sentence level. Instead of predicting a word's context, they predict whole sentences. Skip-thought works as a sequence-to-sequence framework. Given a tuple of consecutive sequences (s_{i-1}, s_i, s_{i+1}) as input, it encodes the considered sentence using a sentence encoder SE in a fixed length vector $h_i = SE(s_i)$. Given the sentence vector, a sentence decoder DE_p aims to generate the previous sentence $DE_p(h_i) = s_{i-1}$ and the next sentence $DE_n(h_i) = s_{i+1}$.

Both the encoder and decoder are trained to maximize the sum of the log-probabilities for the forward and backward sentences conditioned on the encoder representation:

$$\mathcal{L} = \sum_t \log P(w_{i+1}^t|w_{i+1}^{<t}, h_i) + \sum_t \log P(w_{i-1}^t|w_{i-1}^{<t}, h_i)$$

The model is trained on the BookCorpus dataset. The original implementation uses a recurrent neural network, with Gated Recurrent Units (Cho et al. 2014) for the encoder and decoder. The skip-thought method has become popular as the method is fully self-supervised and does not require any labeled data. Moreover, the original paper trained models at scale and released them in open-source.[8]

8: https://github.com/ryankiros/skip-thoughts. Ba, J. R. Kiros, and Hinton (2016) also proposed an upgrade of the model by adding layer normalization.

The method suffers from practical limitations. In particular, it is computationally costly as, in addition to the encoder, it also requires an extra decoder that converts vectors into sentences. Although not used during inference, the decoder part is computationally costly as it requires decoding the words of target sentences sequentially. Each word prediction requires a heavy softmax operation over the entire vocabulary. Overall, it takes two weeks to train the original model.

Sequential Denoising Autoencoder (SDAE) Hill, Cho, and Korhonen (2016) propose a model based on denoising autoencoders (DAEs) for text. The model uses an encoder-decoder framework to reconstruct a corrupted version of the current sentence. As with Skip-thought, the model has a self-supervised objective, but does not require that the training corpus maintains the narrative order of the sequences. The input sentence s is corrupted using a noise function $N(s|p_o, p_x)$ which acts as follows: for each word $w \in s$, N deletes w with (independent) probability p_o. Then, for each non-overlapping bigram $w_i w_{i+1} \in s$, N swaps w_i and w_{i+1} with probability p_x. The encoder-decoder architecture is based on LSTMs and is also trained on the BookCorpus dataset to optimize the following loss function:

$$\mathscr{L} = \sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, h_i)$$

FastSent The method, also introduced in Hill, Cho, and Korhonen (2016), is an additive (log-linear) version of Skip-thought, which aims to lower its computational expense. Given a BoW representation of a considered sentence, the model is trained to predict the words appearing in the context (and optionally, the considered) sentences.

FastSent learns a source u_w and target v_w embeddings for each word in the model vocabulary. Given a tuple of consecutive sequences (s_{i-1}, s_i, s_{i+1}) as input, it encodes the considered sentence as the sum of its word embeddings $h_i = \sum_{w \in s_i} u_w$. Given the representation of the considered sentence, it aims at predicting the words of the context sentences.

$$\mathscr{L} = \sum_{w \in s_{i-1} \cup s_{i+1}} \log P(w|h_i)$$

With $P(w|h_i) = \frac{e^{h_i u_w}}{\sum_{v \in V} e^{h_i u_v}}$. A variant includes the prediction of the words from the considered sentence in addition to those of adjacent sentences. The objective function thus becomes:

$$\mathscr{L} = \sum_{w \in s_{i-1} \cup s_i \cup s_{i+1}} \log P(w|h_i)$$

Quickthought (QT) Logeswaran and Honglak Lee (2018) circumvent some practical limits of Skip-thought by directly operating in the space of sentence embeddings. It uses a discriminative rather than a generative objective.[9] A classifier aims at distinguishing the correct embedding of a target sentence given a set of candidate sentences. The method thus avoids reconstructing the surface form of the input sentence or its neighbors.

9: More broadly, the approach relates to contrastive learning, which is successfully applied in a variety of domains including audio (Oord, Y. Li, and Vinyals 2018), image Tian, Krishnan, and Isola (2020) and Z. Wu et al. (2018), video or word with the negative sampling methods from *word2vec* (Mikolov, K. Chen, et al. 2013; Mikolov, Sutskever, et al. 2013). Some mathematical foundations are detailed in Saunshi et al. (2019)

The method takes inspiration from the distributional hypothesis successfully applied for words, but this time, to identify context sentences. Given a sentence s, a corresponding context sentence s^+ and a set of K negative samples $s_1^- \cdots s_K^-$, the training objective is to maximize the probability to discriminate the correct sentence among negative samples: $p(s^+|s, s_1^- \cdots s_K^-)$. The algorithm architecture used to estimate p is close to *word2vec*. Two sentence encoders f and g are defined, and the conditional probability is estimated as follows:

$$p(s^+|s, s_1^- \cdots s_K^-) = \frac{e^{f(s)^\top g(s^+)}}{e^{f(s)^\top g(s^+)} + \sum_{i=1}^{N} e^{f(s)^\top g(s_i^-)}}$$

The parameters from f and g are trained to maximize the probability of identifying the correct context sentences for each sentence in the training data D:

$$\mathscr{L} = \sum_{s \in D} \log P(s^+|s, s_1^- \cdots s_K^-)$$

The model is also trained on the BookCorpus dataset. Each batch is composed of contiguous sentences from the corpus. All the sentences in the batch constitute the candidates for classification for each sentence. The pre-trained model is also available in open-source.[10] At inference time, the sentence representation is obtained as the concatenation of the two

10: https://github.com/lajanugen/S2V

encoders f and g such as $s \rightarrow [f(s); g(s)]$. f and g are chosen identical and consist of two LSTM.

3.3.3 Multi-task learning

Some frameworks propose to combine the training objective mentioned above in a multi-task setup. We expect the model to encode complementary properties and inductive biases required for each sub-task. Thus, training on many weakly related tasks is expected to improve generalization to novel ones.

The universal sentence encoder (USE) (Cer et al. 2018) trains a transformer and Deep Averaging Network (DAN) on a multi-task setup: a skip-thought objective (R. Kiros et al. 2015), a conversational response prediction, and a supervised natural language inference classification task on the SNLI dataset (Bowman, Angeli, et al. 2015; Conneau, Kiela, et al. 2017).

Subramanian et al. (2018) also propose a multitask learning framework that trains a single model with multiple distinct objectives: context sentences generation (Section 3.3.2), neural machine translation, constituency parsing and natural language inference (Section 3.3.1).

3.3.4 Cross-lingual sentence embeddings

The majority of work presented previously focuses on English. Many approaches, in particular semi-supervised, may be generalized to other languages. An alternative would be to learn a *one-for-all* multilingual model that encompasses a whole family of languages instead of learning separate models for each language. Languages with limited resources may benefit from collaborative training across multiple languages. The ability to represent sentences from different languages within the same representational space may also facilitate the zero-shot transfer between languages. The methods for training such cross-lingual representations are inherently close to mono-lingual ones. We review the more recent approaches below.

LASER is a method for embedding sentences which covers 93 languages (Artetxe and Schwenk 2019). The method has gone through several iterations. The first version is close to Skip-thought (R. Kiros et al. 2015). Skip-thought will generate the previous and next sentence when given a source sentence, while LASER will translate it into a target language. However, there are similarities between the training processes; an encoder SE maps the source sentence into a fixed length vector $h_l = SE(s_l)$. Given the sentence vector, a sentence decoder DE aims to generate the translation in a target language $DE(h_l) = s_k$. We discard the decoder at inference and keep the encoder to embed sentences in any of the training languages. We train both the encoder and decoder to maximize the sum of the log-probabilities for the target sentences conditioned on the encoder representation:

$$\mathcal{L} = \sum_t \log P(w_k^t | w_k^{<t}, h_l)$$

It is important to note that the system relies on a single encoder and decoder, shared for every language. Additionally, the encoder does not receive any explicit information regarding the language of the input, allowing it to learn language-independent representations. The encoder consists of a BiLSTM encoder with 1 to 5 layers, each 512-dimensional. Sentence embeddings are obtained by applying a max-pooling operation over the output of a BiLSTM encoder. The method covers 93 languages in total. However, during training, the method is limited to English and Spanish as target languages because the vast majority of data is aligned with these languages, and considering all possible language pairs would be intractable. The training data includes 223 million parallel sentences from multiple sources available on the OPUS Web site (Tiedemann 2012).

Laser was recently improved into LASER 2 & 3 (Heffernan, Çelebi, and Schwenk 2022). With the new iteration, the authors added 50 low-resource African languages and generally improved the performance. The update moves away from the *one-for-all* but instead learns multiple models for different families of languages. The architecture of the model also evolves toward a 12-layer transformer. The training method relies on the distillation approach from Reimers and Gurevych (2020), which allows extending an existing sentence embedding space to new languages. This supervised teacher-student

approach compares the teacher's sentence representation against the student's target language sentence representation in order to extend the embedding space.

LaBSE is also a cross-lingual sentence embedding method (Feng et al. 2022). Similar to LASER, it aims to translate sentences from a source into a target language. However, it uses a discriminative rather than a generative objective. A translation ranking loss directly maximizes the similarity of translation pairs in a shared embedding space. The method uses a dual encoder, which encodes the source and target sentences separately. Given a batch of N sentences s_l^i, and their translations into a target language s_k^i, the training objective is to maximize the probability to associate each sentence with its true translation over all $N-1$ alternatives in the same batch: $p(s_k^i|s_l^i, s_k^1 \cdots s_k^N)$. The conditional probability is estimated as follows:

$$p(s_k^i|s_l^i, s_k^1 \cdots s_k^N) = \frac{e^{f(s_l^i)^\top g(s_k^i)}}{e^{f(s_l^i)^\top g(s_k^i)} + \sum_{j=1, j\neq i}^{N} e^{f(s_l^i)^\top g(s_k^j)}}$$

The parameters from the two parts of the dual encoder, f and g are trained to maximize the probability of identifying the correct translation for each sentence in the training batch:

$$\mathscr{L} = \sum_{i \in [\![1,N]\!]} \log p(s_k^i|s_l^i, s_k^1 \cdots s_k^N)$$

The approach presents similarities with Quickthought (Logeswaran and Honglak Lee 2018) and Sentence-Bert (Reimers and Gurevych 2019). The authors enhance the method by systematically exploring the combination with the best existing methods for learning sentence embeddings. Since the loss $\mathscr{L}$ is asymmetric and depends on the direction to which the softmax is applied (over the source or the target sentences), the authors use the sum of the source-to-target, $\mathscr{L}$, and target-to-source, $\mathscr{L}'$, losses (Yang, Ábrego, et al. 2019):

$$\bar{\mathscr{L}} = \mathscr{L} + \mathscr{L}' \tag{3.40}$$

The method also implements an additive margin softmax, which introduces a margin m around positive pairs to improve the separation between translations and nearby non-translations (Yang, Ábrego, et al. 2019). The final conditional probability is thus estimated as follows:

$$p(s_k^i|s_l^i, s_k^1 \cdots s_k^N) = \frac{e^{f(s_l^i)^\top g(s_k^i)-m}}{e^{f(s_l^i)^\top g(s_k^i)-m} + \sum_{j=1,j\neq i}^{N} e^{f(s_l^i)^\top g(s_k^j)}}$$

Finally, contrastive learning benefits from large training batch sizes (T. Chen et al. 2020; Guo et al. 2018; Qu et al. 2021) and the authors introduce a method called *cross-accelerator negative sampling*, which allows to better distribute the softmax computation across multiple cores and achieve larger batch size.

LaBSE uses transformer encoders and the l2 normalized `[CLS]` token representations from the last transformer block as sentence embedding. For the pre-training, they collected monolingual 17B monolingual sentences from CommonCrawl[11] and Wikipedia.[12] They pre-trained the model using a masked language model (mlm) (Devlin et al. 2019) and translation language model (tlm) (Conneau and Lample 2019) task. They finalize the pre-training of the models using the translation ranking task with in-batch negative sampling on 6B translation pairs from web pages.

11: https://commoncrawl.org/

12: https://dumps.wikimedia.org/frwiki/

3.4 Evaluating sentence embeddings

Evaluation of sentence embeddings is not a straightforward process. As for the training step, we do not have access to *gold* labels to evaluate our embeddings. We must therefore rely on indirect evaluation methods. The first set of methods in Section 3.4.1 characterizes the quality of the sentence representations given the performance they allow on a task of interest. The second set of methods probes for controlled and targeted linguistic characteristics by the mean or indirect classification tasks on dedicated artificial datasets (Section 3.4.2). Finally, we enumerate in Section 3.4.3 methods to directly analyze the underlying dynamics and mechanisms within the connections of model layers.

3.4.1 Downstream tasks

The SentEval benchmark (Conneau and Kiela 2018) is specifically designed to assess the quality of the embeddings.[13] Each task is formatted as a classification task that takes sentence embeddings as input features. The downstream model usually consists of a simple multi-layer perceptron or logistic regression. It is kept as minimal as possible to avoid the case where uninformative embeddings are compensated by an excellent classifier.[14] Another important reason for using simple downstream classifiers is to assess the straightforward extractability of information from embeddings. Our goal is to identify what information is captured in the embedding vectors rather than assessing whether we can reconstruct the information from the embeddings. The downstream evaluation methods are completely agnostic with respect to the sentence embedding method.[15] The development set is used for each task to choose the regularization parameters, and results are reported on the test set. The tasks include sentiment and subjectivity analysis (**MR, CR, SUBJ, MPQA**), question type classification (**TREC**), paraphrase identification (**MRPC**) and semantic relatedness (**SICK-R**). We give examples for each task in Table 3.3. The **MR, CR, SUBJ, MPQA** and **TREC** are classification tasks, for which we report the accuracy. For **MRPC**, we report the accuracy and f1 score. Finally, the **SICK-R** task is a regression task. We report the Pearson (r) and Spearman (ρ) correlations as well as the mean squared error (mse).

We report in Table 3.4 the downstream results using the training methods enumerated in Section 3.3. We divided the methods into three categories based on the training objective: self-supervised, supervised or semi-supervised, and pre-trained. Not one of models outperforms the others across all tasks. In addition, although Bert is the cornerstone of many NLP applications, its application to this sentence embedding benchmark falls below the state-of-the-art.

However, downstream tasks may suffer from empirical limitations. The downstream performance may not necessarily reflect the quality of the representations. First, the complexity of the tasks makes it difficult to determine what information is captured in the representations. Then, uncontrolled effects can inflate the perception of success on downstream tasks: certain hyper-parameters such as the embedding dimensions

13: Senteval is posterior to most of the references. However, these studies do evaluate on tasks later included in the benchmark.

14: It is important to make the distinction between the training of the sentence embedding methods (detailed in Section 3.3 and the training of the downstream classifier which uses the sentence embedding as input but doesn't further train them.

15: Contrary to GLUE and SuperGLUE benchmarks (A. Wang, Pruksachatkun, et al. 2019; A. Wang, Singh, et al. 2019), the sentence embedding model is not fine-tuned during the evaluation. We specifically evaluate the information within sentence embeddings and not the model used to produce them.

Table 3.3: Examples from the tasks of the SentEval benchmark that we will use in our experiments. *N* is the number of samples. MR, CR, SUBJ, and MPQA are binary classification tasks with labels positive or negative. SICK-R is a Semantic Textual Similarity (STS) task for which labels are scores between 0 and 5. MRPC is a Paraphrase Detection (PD) task for which labels are true or False. TREC is a 6-class classification problem. We adapted this example table from Conneau and Kiela (2018).

Task	N	Sentence 1	Sentence 2	Label
MR	11k	"Too slow for a younger crowd , too shallow for an older one."		neg
CR	4k	"We tried it out christmas night and it worked great ."		pos
SUBJ	10k	"A movie that doesn't aim too high , but doesn't need to.		subj
MPQA	11k	"don't want"; "would like to tell";		neg, pos
TREC	6k	"What are the twin cities ?"		LOC:city
SICK-R	10k	"A man is singing a song and playing the guitar"	"A man is opening a package that contains headphones"	1.6
MRPC	5.7k	"The procedure is generally performed in the second or third trimester."	"The technique is used during the second and, occasionally, third trimester of pregnancy."	paraphrase

may impact downstream performance; models may also exploit superficial cues or structural biases from the evaluation datasets.

In that regard, Wieting and Kiela (2019) propose a rather disturbing study in which they test randomly initialized encoders on downstream tasks and still obtain competitive results. They show that many parameters impact downstream performance above and beyond the encoder structure. In particular, the quality of the word embeddings being composed by the encoder, or the dimension of the word and sentence embeddings. Similarly, Adi et al. (2017) demonstrate that a BoW composition model is 70% accurate on a binary word orders prediction task. Since BoW model does not preserve word order information, Ettinger et al. (2018) interpret that the above-chance performance appears to rely on statistical regularities of word ordering in the train and test sets.

3.4.2 Probing tasks

Probing tasks evaluate representations on a per-phenomenon basis. They aim to determine which precise semantic, syntactic, lexical, or surface information of the input sentence is captured in its embeddings. They typically consist of simple classification tasks contingent on a precise linguistic property,

Table 3.4: SentEval task results using fixed sentence encoder. We divided the table into sections. The first range of models uses self-supervised training objective. The second section present models trained on labelled or semi-automatically labeled data. The third section reports pre-trained transformers based-models. FastSent is reported from Hill, Cho, and Korhonen (2016). Skipthoughts results from (R. Kiros et al. 2015) Skipthoughts + LN which includes layer normalization method from Ba, J. R. Kiros, and Hinton (2016). We considered the Quickthought results (Logeswaran and Honglak Lee 2018). DisSent and Infersent are reported from A. Nie, Bennett, and Goodman (2019) and Conneau, Kiela, et al. (2017) respectively. Pre-trained transformers results are reported from Reimers and Gurevych (2019). We report laBSE results from (Feng et al. 2022). We run the evaluation on SentEval for LASER 1 and 2 respectively (Artetxe and Schwenk 2019; Heffernan, Çelebi, and Schwenk 2022). The **Hrs** column indicates indicative training time, the **Dim** column corresponds to the sentence embedding dimension. †, indicates models that we had to re-train. Best results in each section are shown in **bold**, best results overall are underlined. Performance for **SICK-R** results are reported by convention as ρ and $r \times 100$.

Model	Dim	Hrs	MR	CR	SUBJ	MPQATREC	MRPC		SICK-R		
							Acc	F1	r	ρ	MSE
Context sentences prediction											
FastSent	≤ 500	2	70.8	78.4	88.7	80.6	76.8	72.2	80.3	—	—	—
FastSent + AE	≤ 500	2	71.8	76.7	88.8	81.5	80.4	71.2	79.1	—	—	—
Skipthought	4800	336	76.5	80.1	93.6	87.1	92.2	73.0	82.0	85.8	79.2	26.9
Skipthought + LN	4800	672	79.4	83.1	93.7	89.3	—	—	—	85.8	78.8	27.0
Quickthought	4800	11	**80.4**	**85.2**	**93.9**	**89.4**	**92.8**	**76.9**	**84.0**	**86.8**	**80.1**	**25.6**
Sentence relations prediction												
InferSent	4096	—	**81.1**	**86.3**	92.4	90.2	88.2	**76.2**	**83.1**	**88.4**	—	—
DisSent Books 5	4096	—	80.2	85.4	93.2	90.2	91.2	76.1	—	84.5	—	—
DisSent Books 8	4096	—	79.8	85.0	**93.4**	**90.5**	**93.0**	76.1	—	85.4	—	—
Pre-trained transformers												
Bert-base [CLS]	768	96	78.7	84.9	94.2	88.2	**91.4**	71.1	—	75.7†	—	—
Bert-base [NLI]	768	96	**83.6**	**89.4**	**94.4**	**89.9**	89.6	**76.0**	—	**84.4**†	—	—
Multi-lingual												
LASER-1†	1024	120	72.4	78.1	90.4	84.1	83.2	72.6	79.0	80.5	75.0	36.0
LASER-2†	1024	120	74.7	77.2	91.1	88.0	81.6	**77.0**	**83.7**	**82.9**	**77.6**	**32.2**
laBSE	768	96	**79.1**	**86.7**	**93.6**	**89.6**	**92.6**	74.4	—	—	—	—

and this targeted approach simplifies interpretations. Probing and downstream evaluation follow the same protocol: the probing classifier takes as input feature the sentence embeddings produced by a given encoder (as for the downstream evaluation, the embeddings are not further tuned in that phase). Therefore, high accuracy on the task should indicate that the information is encoded in the input embeddings.

Probing tasks require maintaining access to the detailed labeling of the linguistic phenomenon of interest. Given this information, it is possible to partition the dataset and decompose the model performance given this specific phenomenon. This partition should also maintain the dataset distribution unchanged regarding any other linguistic phenomena and

remove any uncontrolled bias toward this specific aspect.[16] Last but not least, it should retain a variety of sentences that will be encountered in natural-occurring text.

16: For example, Bentivogli et al. (2016) and Lai and Hockenmaier (2014) observed structural biases in the SICK (Marelli et al. 2014) dataset distribution. As a consequence, a simple heuristic detecting negation is sufficient to achieve high accuracy for the textual entailment task.

Probing tasks must therefore be constructed in a rigorous and controlled manner. The dataset is usually created either by labeling natural occurring sentences or by semi-automatically generating sentences that follow specific properties. The first method facilitates access to a wide variety of syntactic structures and configurations. On the other hand, semi-automatically sentence generation allows for the precise control of their targeted characteristics.

Probing task is an active subject of research and has been adapted for many linguistic properties. Conneau, Kruszewski, et al. (2018) aggregate 10 tasks—including those introduced in Adi et al. (2017)—in a benchmark. The tasks test for surface, semantic and syntactic information. The sentence length (**SentLen**) task aims at predicting the length of sentences in terms of word number. The word content (**WC**) task determines the ability to recover the original words in a sentence from the embedding. The bigram shift (**BShift**) tests the sensitivity to original word orders. The tree depth (**TreeDepth**) aims at predicting the depth from a hierarchical sentence structure. The top constituent task (**TopConst**), aims at predicting the top constituents immediately below the sentence root node. The **Tense** task aims at predicting the tense of the main clause verb. The subject and object number (respectively **SubjNum** and **ObjNum**) tasks focus on the number of respectively the subject and object of the main clause. The semantic odd man out (**SOMO**) task aims at identifying sentences for which random nouns or verbs were replaced. Finally, the coordination inversion (**CoordInv**) aims at identifying sentences for which the order of the clauses was or not modified.

3.4.3 Analysis of the internal dynamics underlying NLP models

Finally, some alternative approaches propose intuitive visualization techniques that allow interpreting neural network mechanisms when processing specific examples. J. Li et al. (2016) propose to analyze compositional model properties with some specific plots. Using dimensionality reduction

methods, they project words or phrases before and after modifying, negating, or composing clauses. Additionally, they display the saliency of individual tokens with respect to their predictions. Other methods propose visualization of neural model hidden states. Strobelt et al. (2018) represent recurrent models hidden states. While Hoover, Strobelt, and Gehrmann (2019) propose a similar tool for the analysis of transformers.

Toward integrating linguistic biases into neural networks

Contrasting distinct structured views to learn sentence embeddings

4

> *When people say AI has "learned x" what they usually mean is that a deep learning model has learned a dataset well enough to find the pattern you asked for. It has no symbolic or logical abstraction. It is kahnemann system 1. It looks smart. It isn't.*
>
> — **Mark Madsen**
> Twitter post, 2019

We hypothesize that structure is a crucial element to perform compositional knowledge. In particular, the heterogeneity of performances across models and tasks makes us assume that some structures may be better adapted for a given example or task. Therefore, combining diverse structures should be more robust for tasks requiring complex word composition to derive their meaning. Hence, we aim to evaluate the potential benefit from interactions between pairs of encoders. In particular, we propose a training method for which distinct encoders are learned jointly. We conjecture this association might improve our embeddings' power of generalization and propose an experimental setup to test our hypothesis.

We take inspiration from multi-view learning, which is successfully applied in a variety of domains. In such a framework, the model learns representations by aligning separate observations of the same object. Such observations are referred to as *views*. In our case, we consider a view for a given sentence as the association of the plain sentence with several kinds of syntactic representations.

Combining different structural views has already been proven to be successful in many NLP applications. Kong and G. Zhou (2011) provide a heuristic to combine dependency and constituency analysis for coreference resolution. Ahmed, Samee, and Mercer (2019) and Y. Zhou, C. Liu, and Pan (2016) combine Tree LSTM and standard sequential LSTM with a cross-attention method and observe improvements on a semantic textual similarity task. L. Chen et al. (2017) combine CNN and Tree LSTM using attention methods and outperform both models taken separately on a sentiment classification

task. Finally, Q. Chen et al. (2017) combine sequential LSTM and Tree LSTM for natural language inference tasks.

The novelty here is to combine distinct structured models to build standalone sentence embeddings, which has not yet been explored. This paradigm benefits from several structural advantages. It pairs nicely with contrastive learning, as already mentioned. It may thus be trained in a self-supervised manner that does not require data annotation. Moreover, our method is not specific to a certain kind of encoder architecture, and it does not require, for example, the use of attention layers or tree-structured models.

Our setup could therefore be extended with any encoding function. Finally, our training method induces an interaction between models during inference and, paramountly, during the training phase.

We organize our section as follows: Section 4.1 reviews the contrastive and multi-view training method we used. In Section 4.2, we present our training and evaluation setup. We then propose an in-depth analysis of our results.

4.1 Method

4.1.1 Contrastive learning

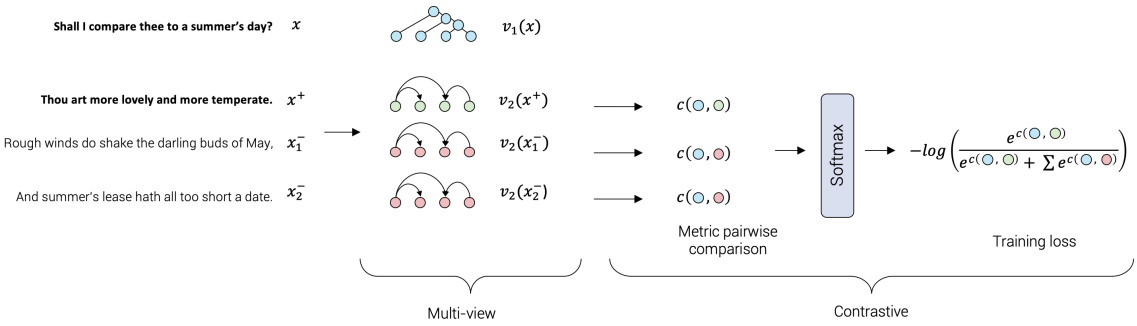


Figure 4.1: Contrastive training method. The objective is to reconstruct the storyline. Sentences are presented in their original order. Given an anchor sentence x, the model has to identify the context sentence x^+ out of negative samples x_1^-, x_2^-. Sentences are encoded using separate views, which are composed within a pairwise distance matrix.

We train our model using the contrastive objective from Logeswaran and Honglak Lee (2018), detailed in Section 3.3. The method takes inspiration from the distributional hypothesis successfully applied for words, but this time, to identify

context sentences. Given a sentence s, a corresponding context sentence s^+ and a set of K negative samples $s_1^- \cdots s_K^-$, the training objective is to maximize the probability of predicting the correct sentence among negative samples: $p(s^+|S)$ with $S = \{s, s^+, s_1^- \cdots s_K^-\}$. As illustrated in Figure 4.1, two sentences encoders f and g are defined and the conditional probability is estimated as follow:

$$p(s^+|S) = \frac{e^{c(f(s),g(s^+))}}{e^{c(f(s),g(s^+))} + \sum_{i=1}^{N} e^{c\left(f(s),g(s_i^-)\right)}}$$

With $c(x, y)$ the scoring function. Logeswaran and Honglak Lee (2018) simply use an inner product for c such as $c\,(x, y) = x^\top y$. In our case, as the encoders f and g have distinct architectures. To prevent the case of f and g having distinct norms and the inner product resulting in irrelevant information, we choose a bilinear function defined as $c\,(x, y) = x^\top W y$ (Tschannen et al. 2020a). At inference time, the sentence representation is obtained as the concatenation of the two encoders f and g such as $s \rightarrow [f(s); g(s)]$, as illustrated in Figure 4.2. In Logeswaran and Honglak Lee (2018), f and g use the same RNN encoder. However, the authors observe that the encoders might learn redundant features. To limit this effect, they propose to use a distinct set of embeddings for each encoder.

We propose addressing this aspect by enhancing the method with a multi-view framework and using a distinct structured model for the encoders f and g. We hypothesize that some structures may be better adapted for a given example or task. For example, dependency parsing usually sets the verb as the root node. Whereas in constituency parsing, subject and verb are often split between the left and right sub-trees from the root node (as illustrated in Figure 3.2). Therefore, the combination of different structures should be more robust for tasks requiring complex word composition and be less sensitive to lexical variations. Consequently, we propose a training procedure that allows the model to benefit from the interaction of various syntactic structures.

4.1.2 Language views

Multi-view aims at learning representations from data represented by multiple independent sets of features. We generalize the notion of view for a sentence as the application of a specific syntactic framework. For each view, we use an *ad-hoc* algorithm that maps the structured sentence into an embedding space.

We consider structures exposed in Section 3.2: Vanilla GRU (Seq), dependency tree combined with an attentive Child-Sum Tree LSTM (Dep), Constituency tree combined with N-Ary Tree LSTM (Const).[1] Although under some hypotheses equivalences might be derived between the last two representations schemes, we hypothesize that, in our context, the corresponding sequence of operations might provide the possibility of capturing rather distinct linguistic properties. The various models may, therefore, be complementary and their combination allows for more fine-grained analysis.

1: We introduced this attentive version of the Child-Sum Tree LSTM for which details are given in Section 3.2.3

For the Dep view, the dependency tree is obtained using the deep biaffine parser from Dozat and Manning (2017). We used an open-source implementation of the parser and replaced the pos-tag features with features obtained with Bert.[2] Therefore we do not need pos-tags annotations to parse our corpus.

2: https://github.com/yzhangcs/biaffine-parser

For the Const view, the structure is obtained using the constituency neural parser from Kitaev and Klein (2018). We binarize the trees to ensure that every node has exactly two dependents. The binarization is performed using a left markovization (Klein and Manning 2003) and unary productions are collapsed in a single node. Regarding the inference speed, the constituency parser is the bottleneck and parses around 500 sentences/second. In our case, the parsing of the entire corpus (40M sentences) takes about a day to complete.

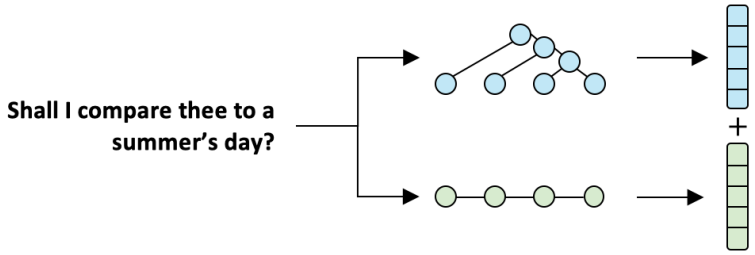


Figure 4.2: Multi-view sentence embedding. At inference, embeddings are the concatenation from both views.

4.2 Experiments

We train our models on the UMBC dataset (Han et al. 2013).[3]* We limit our corpus to the first 40M sentences from the tokenized corpus. Indeed, Logeswaran and Honglak Lee (2018) already analyzed the effect of the corpus size, and we focus here on the impact of our multi-view setting.[4] For each sample, we train the model to maximize the probability of predicting the correct sample among negative samples. In practice, we constitute mini-batches of consecutive sentences. For each sentence in the mini-batch, the correct samples correspond to context sentences, that is, sentences immediately after and before the target sentence. Other sentences in the batch are considered negative examples. We use a batch size of 400. Therefore, for each target sentence, we consider 2 positive samples and 397 negative samples. Model hyper parameters are fixed given literature on comparable work (Logeswaran and Honglak Lee 2018; Tai, Socher, and Manning 2015). All models are trained using the Adam optimizer with a $5e^{-4}$ learning rate. Regarding the infrastructure, we use a Nvidia GTX 1080 Ti GPU. All model weights are initialized with a Xavier distribution and biases set to 0. We do not apply any dropout.

For the vocabulary, we follow the setup proposed in R. Kiros et al. (2015) and Logeswaran and Honglak Lee (2018) and we train two models in each configuration. We train a first model initialized with pre-trained embedding vectors and do not update them during training. The vocabulary includes the top 2M cased words from the 300-dimensional GloVe vectors (Pennington, Socher, and Manning 2014).[5] We train another model limited to a 50K words vocabulary, randomly initialized with a Xavier distribution and updated during training. At inference, the vocabulary is expanded to 2M words using a linear projection.

3: The bookcorpus introduced in Zhu et al. (2015) and traditionally used for sentence embedding is no longer distributed for copyright reasons. Therefore, we prefer a corpus freely available. The impact of the training dataset choice is analyzed in Section 4.2.4.

4: Logeswaran and Honglak Lee (2018) improve the average result on SentEval from 0.7 points by increasing the coprus size from 45M to 174M sentences.

5: https://nlp.stanford.edu/projects/glove/

4.2.1 Evaluation on SentEval

As is usual for models aiming to build generic sentence embeddings (Arora, Y. Liang, and Ma 2017; Conneau, Kiela, et al. 2017; Hill, Cho, and Korhonen 2016; R. Kiros et al. 2015;

* https://ebiquity.umbc.edu/blogger/2013/05/01/umbc-webbase-corpus-of-3b-english-words/

Table 4.1: SentEval Task Results Using Fixed Sentence Encoder. We divided the table into sections. The first range of models is directly comparable to our model as the training objective is to identify context sentences. The second section objective is to identify the correct relationship between a pair of sentences. The third section reports pre-trained transformer-based models. The last section reports the results from our models. FastSent is reported from Hill, Cho, and Korhonen (2016). Skipthoughts results from R. Kiros et al. (2015) Skipthoughts + LN which includes layer normalization method from Ba, J. R. Kiros, and Hinton (2016). We considered the Quickthought results Logeswaran and Honglak Lee 2018 with a pre-training on the bookcorpus dataset. DisSent and Infersent are reported from A. Nie, Bennett, and Goodman (2019) and Conneau, Kiela, et al. (2017) respectively. Pre-trained transformers results are reported from Reimers and Gurevych (2019). The **Hrs** column indicates indicative training time, the **Dim** column corresponds to the sentence embedding dimension. † indicates models that we had to re-train. Best results in each section are shown in **bold**, best results overall are underlined. Performance for **SICK-R** results are reported by convention as ρ and $r \times 100$. In all columns, higher scores indicate better performance, with the exception of the MSE, in which lower results indicate better performance

Model	Dim	Hrs	MR	CR	SUBJ	MPQATREC	MRPC		SICK-R		
							Acc	F1	r	ρ	MSE
Context sentences prediction											
FastSent	≤500	2	70.8	78.4	88.7	80.6	76.8	72.2	80.3	—	—	—
FastSent + AE	≤500	2	71.8	76.7	88.8	81.5	80.4	71.2	79.1	—	—	—
Skipthought	4,800	336	76.5	80.1	93.6	87.1	92.2	73.0	82.0	85.8	79.2	26.9
Skipthought + LN	4,800	672	79.4	83.1	93.7	89.3	—	—	—	85.8	78.8	27.0
Quickthought	4,800	11	**80.4**	**85.2**	**93.9**	**89.4**	**92.8**	**76.9**	**84.0**	**86.8**	**80.1**	**25.6**
Sentence relations prediction											
InferSent	4,096	—	**81.1**	**86.3**	92.4	90.2	88.2	**76.2**	**83.1**	**88.4**	—	—
DisSent Books 5	4,096	—	80.2	85.4	93.2	90.2	91.2	76.1	—	84.5	—	—
DisSent Books 8	4,096	—	79.8	85.0	**93.4**	**90.5**	**93.0**	76.1	—	85.4	—	—
Pre-trained transformers											
Bert-base [CLS]	768	96	78.7	84.9	94.2	88.2	**91.4**	71.1	—	75.7†	—	—
Bert-base [NLI]	768	96	**83.6**	**89.4**	**94.4**	**89.9**	89.6	**76.0**	—	**84.4**†	—	—
Our models (GloVe & Pretrained Embeddings)											
Seq, Const†	4,800	41	79.8	82.9	94.6	88.5	90.4	76.4	83.7	86.1	78.9	26.3
Dep, Seq†	4,800	27	79.7	82.2	94.4	88.6	91.0	**77.9**	**84.4**	86.6	79.8	25.5
Dep, Const†	4,800	39	**80.7**	**83.6**	**94.9**	**89.2**	**92.6**	76.8	83.6	**87.0**	**80.3**	**24.8**

6: Senteval is posterior to most of the references. However, these studies do evaluate on tasks later included in the benchmark.

Logeswaran and Honglak Lee 2018; A. Nie, Bennett, and Goodman 2019), we use tasks from the SentEval benchmark (Conneau and Kiela 2018).[6] SentEval is specifically designed to assess the quality of the embeddings themselves rather than the quality of a model specifically targeting a downstream task, as is the case for the GLUE and SuperGLUE benchmarks (A. Wang, Pruksachatkun, et al. 2019; A. Wang, Singh, et al. 2019). Indeed, the evaluation protocol prevents fine-tuning the model during inference and the architecture to tackle the downstream tasks is kept minimal. Moreover, the embedding is kept identical for all tasks, thus assessing their properties of generalization.

Therefore, classification tasks from the SentEval benchmark

are usually used for evaluation of sentence representations (Conneau and Kiela 2018): the tasks (presented in Section 3.4.1) include sentiment and subjectivity analysis (**MR, CR, SUBJ, MPQA**), question type classification (**TREC**), paraphrase identification (**MRPC**) and semantic relatedness (**SICK-R**). Contrasting the results of our model on this set of tasks will help to better understand its properties. MR, CR, SUBJ, MPQA tasks are binary classification tasks with no predefined train-test split. We therefore use a 10-fold cross validation. For the other tasks, we use the proposed train/dev/test splits. We give examples for each tasks in Table 3.3. We follow the linear evaluation protocol of R. Kiros et al. (2015), where a logistic regression or softmax classifier is trained on top of sentence representations. The dev set is used to choose the regularization parameters and results are reported on the test set.

The **MR, CR, SUBJ, MPQA** and **TREC** are classification tasks, for which we report the accuracy. For **MRPC**, we report the accuracy and f1 score. Finally, the **SICK-R** task is a regression task. We report the Pearson (r) and Spearman (ρ) correlations as well as the mean squared error (mse).

We compare the properties of distinct views combination on downstream tasks. Results are compared with state-of-the-art methods in Table 4.1. The first set of methods (*Context sentences prediction*) are trained to reconstruct the storyline of books. The second set of models (*Sentence relations prediction*) is pre-trained on a supervised task. Infersent (Conneau, Kiela, et al. 2017) is trained on the SNLI dataset, which proposes to predict the entailment relation between two sentences. DisSent (A. Nie, Bennett, and Goodman 2019) proposes a generalization of the method and builds a corpus of sentence pairs with more possible relations between them. Finally, we include models relying on transformer architectures (Pre-trained transformers) for comparison. In particular, Bert-base and Bert-base fine-tuned on the SNLI dataset (Reimers and Gurevych 2019). In Table 4.1, we observe that our models expressing a combination of views such as (Dep, Seq) or (Dep, const) give better results than the use of the same view (seq, seq). It seems that the entanglement of views benefits the sentence embedding properties. In particular, we obtain state-of-the-art results for almost every metric from **MRPC** and **SICK-R** tasks, which focus on paraphrase identification. For the **MRPC** task, we gain a full point in accuracy and out-

perform BERT models. We hypothesize structure is important for achieving this task, especially as the dataset is composed of rather long sentences. The **SICK-R** dataset is structurally designed to discriminate models that rely on compositional operations.

This also explains the score improvement on this task. Tasks such as **MR**, **CR** or **MPQA** consist in sentiment or subjectivity analysis. We hypothesize that our models are less relevant in this case: such tasks are less sensitive to structure and depend more on individual word or lexical variation.

4.2.2 Impact of the multi-view

We aim to measure the impact of multi-view specifically. Table 4.2 compares all possible view pairs out of DEP, CONST and SEQ views. For each multi-view model, we report the average score from SentEval tasks.[7] The first section of the Table corresponds to *single-view* models, for which both views from the pair are identical. The second section reports multi-view models.

7: We scale all metrics as percentages. In particular, we use 100 - MSE for the **SICK-R** task. The final score corresponds to the average of all tasks. We average the scores for tasks with multiple metrics (**MRPC** and **SICK-R**).

Multi-view models outperform those using a single view. Given our experiment, it is advantageous to use multiple views instead of one. It also confirms our hypothesis that combining multiple structured models or views yields richer sentence embeddings.

Table 4.2: Impact of the multi-view. The first section corresponds to single-view setups for which f and g are the same views. The second section reports multi-view models. For each model, we report the average score on the SentEval benchmark.

Model	Dim	Avg. SentEval Score
Single-view models		
CONST, CONST	4,800	84.4
DEP, DEP	4,800	84.6
SEQ, SEQ	4,800	84.9
Multi-view models		
SEQ, CONST	4,800	85.1
SEQ, DEP	4,800	85.3
DEP, CONST	4,800	**86.0**

4.2.3 Qualitative results

We analyze the embeddings from a qualitative perspective and explore the sentences from the **SICK-R** test set. We retrieved the closest neighbors using cosine distance. We

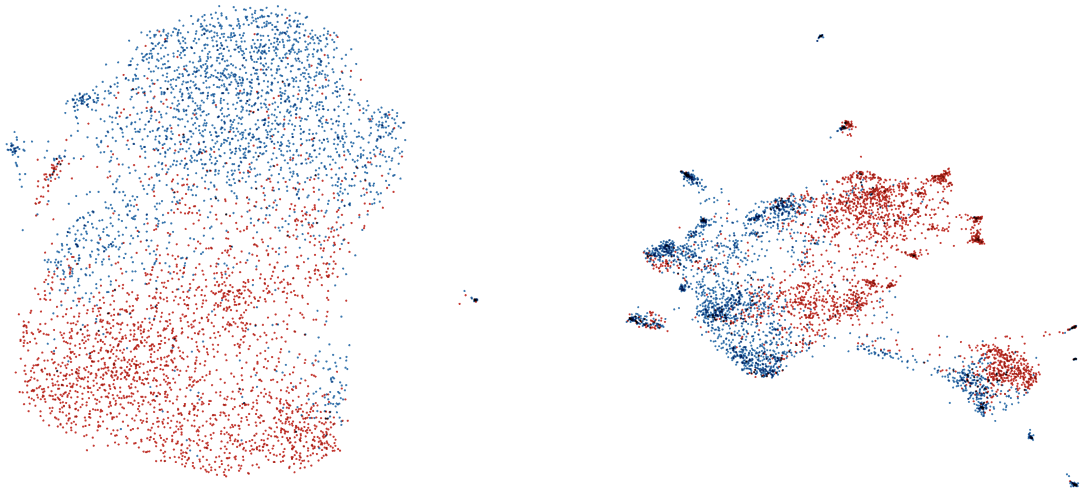


Figure 4.3: Projection of the embeddings from the **SUBJ** task. *(left)* The Dep, Const model is used (*right*) We train a the Quickthought model using scripts from (Logeswaran and Honglak Lee 2018) on the UMBC dataset. Both dimension reductions are performed using the UMAP algorithm (McInnes and Healy 2018). Points in blue correspond to sentences with the label "objective". Points in red correspond to sentences with the label "subjective". In both cases, samples appear well separated given their labels.

compare the results with the Quickthought model. We illustrate in Table 4.3 a panel of examples presenting interesting linguistic properties. Models seem somehow robust to adjective expansions illustrated in the first examples. Indeed, the closest expression from *"A black bird "* is *"A bird , which is black"*. However, the second retrieved sentence is semantically correct for only the Const, Seq association. Quick-thought and Dep, Const present a weakness toward word scrambling for this specific example. We investigate passive forms in the second example. The Const, Seq and Quickthought models seem to attach too much weight to the sentence syntax rather than the semantic. This time the association of Dep and Const views retrieves corresponding active sentences. Last but not least, we examine how models respond to the notion of scaling. Interestingly, Quickthought and Dep, Const are able to bring together "*crowd*" and "*group*" notions.

From a graphic perspective, we projected in two dimensions the sentences from the **SUBJ** task, for which we obtained state-of-the-art results. We use the UMAP (McInnes and Healy 2018) algorithm for dimensionality reduction and compare our multi-view setup with the Quickthought model. The projection is illustrated in Figure 4.3. While the Figure does not reveal any critical distinction between models, samples appear well separated in both cases.

Table 4.3: A qualitative exploration of the sentence embedding space. We embed the sentences from the **SICK-R** test set. Given a query sentence, we retrieve the closest two sentences from the dataset using cosine distance. We compare the results of the semantic search using distinct views or single views combinations.

Encoder	Query and two closest sentences	Cosine distance
	A black bird is sitting on a dead tree	
Dep, Const	A bird , which is black , is sitting on a dead tree	0.118
	A dead bird is near a black man sitting on a tree	0.139
Const, Seq	A bird , which is black , is sitting on a dead tree	0.118
	The black bird is sitting in a leafless tree	0.143
Quickthought	A bird , which is black , is sitting on a dead tree	0.172
	A dead bird is near a black man sitting on a tree	0.172
	Rugby is being played by some men	
Dep, Const	Rugby players are tackling each other	0.381
	Some men are playing rugby	0.392
Const, Seq	Guitar is being played by two men	0.401
	Rugby players are tackling each other	0.403
Quickthought	Guitar is being played by two men	0.455
	Rugby players are tackling each other	0.462
	A crowd of people is near the water	
Dep, Const	A crowd of people is far from the water	0.079
	A group of people is near the ocean	0.356
Const, Seq	A crowd of people is far from the water	0.063
	A man is coming out of the water	0.313
Quickthought	A crowd of people is far from the water	0.067
	Two people are wading through the water	0.388

4.2.4 Impact of the corpus choice

We choose to make use of a distinct corpus as the BookCorpus dataset is no longer distributed for copyright reasons. We run QuickThought scripts (Logeswaran and Honglak Lee 2018) using our dataset based on the UMBC corpus to compare both setups. Results are detailed in the first section from Table 4.4 and are rather close in both configurations. Indeed, except for the **SUBJ** and **MR** task, the use of our dataset penalizes the results. Regarding the dataset size and the SentEval results, we have considered that the comparison holds.[8]

4.2.5 Biases toward embedding size

As exposed in Section 3.3.1, SentEval evaluation framework is suspected to suffers from biases toward the embedding

Model	Dim	Avg. SentEval Score
Dep, Const† (our model)	4,800	86.0
Impact of the pretraining corpus on Quickthought		
Quickthought (results from paper)	4,800	86.1
Quickthought (UMCB 40M)†	4,800	86.2
Impact of the embedding size		
Bert-base [CLS]†	768	78.2
Bert-base [CLS] /w random projection†	4,096	80.3
Impact of pre-training		
Rand†	4,096	45.3
Rand BoW†	4,096	63.7
Rand LSTM	4,096	83.4

Table 4.4: Study on SentEval task results. We first report our average score on the benchmark with our multi-view Dep, Const model. We compare distinct configurations with this reference to better qualify specific parameters' impact. The first section compares the impact of the training dataset for Quickthought. The following section focuses on the impact of the embedding size. To this end, hidden representations are projected into a larger embedding space using a random, fully connected layer. The final section compares an entirely random projection and a BoW or LSTM models randomly initialized with those pretrained on our self-supervised task. † indicates models that we train.

size (Eger, Rücklé, and Gurevych 2019). In addition, some studies suggest that random initialization of encoders may yield surprisingly good results (Wieting and Kiela 2019). We provide extra analysis to discuss these potential pitfalls.

Regarding the dependency on the embedding size, we run experiments to analyze if such bias could explain Bert low performance on SentEval since the output hidden size is only of 768. Following the protocol from Wieting and Kiela (2019), we project the embedding from the CLS token using a random matrix initialized with a glorot distribution. This setup expands Bert embedding into 4,096 dimensions. We reported the results in Table 4.4. Using this random projection, it appears semantic information is not lost. On the contrary, we observe that expanding the embedding size seems to slightly improve the results. However, the results are still below Quickthought vectors by a large margin.

Wieting and Kiela (2019) observe that randomly initialized encoders achieve surprisingly good results on SentEval. We reported the average score from a randomly initialized LSTM in Table 4.4. First, we should note that, while randomly initialized encoders yield surprisingly good results, they are still below the results obtained by pre-training. Nevertheless, we have run additional experiments to better understand this surprising outcome. We present the average scores achieved

8: There are other minor distinctions with the original QuickTought experiment which explains the gap between the (seq, seq) configuration in Table 4.2 and the reproduction of QuickThought results with the original implementation in Table 4.4. These distinctions include: weight decay (1e-4 with the adam optimizer vs. no decay), a bilinear critic to measure the similarity between vectors x and y (i.e., $x^\top W y$ instead of $x^\top y$), a slightly slower vocabulary size (2M instead of 2.1M), a cased sensitive vocabulary instead of lowercase, a slightly different initialization range (uniform between -0.005 and 0.005 vs. uniform between -0.1 and 0.1), a different corpus (which should be marginal as discussed in this section), and slightly different evaluation scripts (different number of layers for the classifier, batch size, tenacity, and number of epochs).

with random sentence embeddings and a BoW model with word embeddings initialized randomly. The results for the entirely random system are below chance. In contrast, the random bag of words lies somewhere between the entirely random and the random LSTM system. While embeddings are randomly initialized, we interpret that BoW can make a partial distinction between words. As a result, the model is capable of capturing, to some degree, lexical information. This phenomenon is likely to occur in the LSTM as well. While the weights for the model are randomly initialized, how the representations are computed allows the model to capture a minimum amount of syntactic information.

4.3 Conclusion and future work

Inspired from linguistic insights and supervised learning, we hypothesize that structure is a central element to build sentence embeddings. The novelty here is detailed in Section 4.1 and consists in jointly learning structured models in a contrastive framework. In Section 4.2 we evaluate the standalone sentence embeddings and use them as a feature for the dedicated SentEval benchmark. We obtain state-of-the-art results on tasks which are expected, by hypothesis, to be more sensitive to sentence structure. We show in Section 4.2.2 that multi-view embeddings yield better downstream task results. Our result confirms our hypothesis that combining diverse structures should be more robust for tasks requiring to perform complex compositional knowledge.

Jointly learning model structure and compositional operations

5

> *Give orange me give eat orange me eat orange give me eat orange give me you.*
>
> — **Nim Chimpsky**
> Male chimpanzee, 1979

This chapter examines the possibility of including tree-like structural bias in neural models while minimizing or eliminating direct structure supervision.

There is this strong hypothesis in computational linguistics that language has a recursive structure (Chomsky 1956). Thus, computing sentence semantic representations traditionally calls for a recursive compositional function whose structure is tree-shaped. As illustrated in Figure 5.1, we can use a sentence structure as support to compute semantic representations—in this case, FOL statements, as introduced in Section 2.3. When using vector representations, we can also use the structure as support to encode the sentence. Following this direction, Socher, Perelygin, et al. (2013) introduce the Stanford sentiment treebank: a corpus with fully labeled parse trees that can be used to analyze the compositional effects of sentiment in language. The dataset provides fine-grained information about lexical units carrying positive or negative sentiment. As illustrated in Figure 5.1, a sentiment prediction system cannot predict the sentiment of a given sentence by simply averaging the sentiment carried by each word. We can only infer this sentiment by analyzing the sentence's structure together with individual words.

Socher, Perelygin, et al. (2013) propose to combine the Stanford sentiment treebank with recursive neural networks. We already introduced such architectures in Section 3.2.3 and successfully used them in Chapter 4. Recursive neural networks represent a phrase using word vectors and a parse tree. They compute parent vectors in a bottom-up fashion using the children as input arguments from a composition function. The composition function is shared across all node computations. Socher, Perelygin, et al. (2013) show that, unlike bag of words, recursive networks can capture the scope

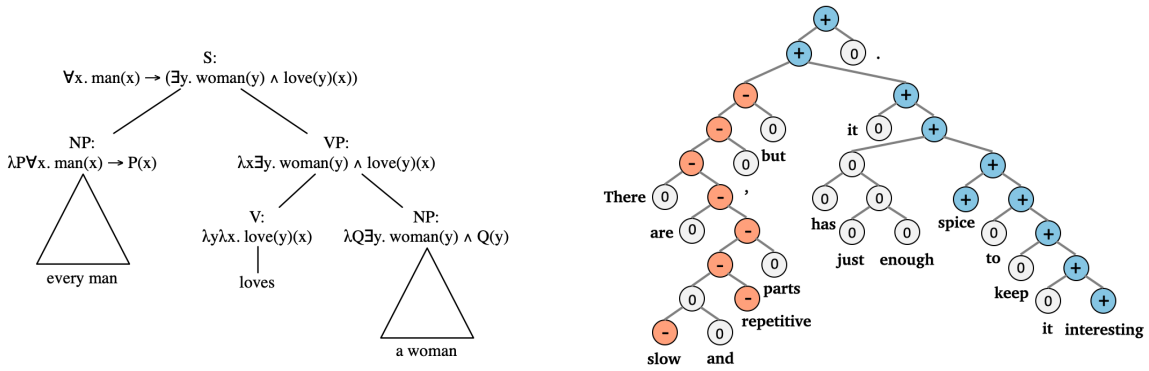


Figure 5.1: *(left)* We illustrate a Montague-style (Montague 1973) derivation of a semantic representation for the sentence "Every man loves a woman." We extracted the figure from https://www.coli.uni-saarland.de/~koller/papers/sem-handbook.pdf. *(right)* We extracted a sentence from the Stanford sentiment treebank. The sample provides annotation for each word contribution to the final sentence sentiment. There are two parts in the sentence "There are slow and repetitive parts, but it has just enough spice to keep it interesting.", respectively negative and positive. The final sentence ends up positive. We adapted the figure from http://nlp.stanford.edu:8080/sentiment/rntnDemo.html.

of negation and sentiment change induced by contrastive conjunctions such as "but".

However, not everyone has access to resources as rich as the Stanford sentiment treebank. The corpus is only available in English and requires precise annotations from experts in linguistic. This chapter investigates the possibility of incorporating tree structural biases with minimal explicit supervision. To this end, we propose a model that jointly parses sentences into discrete trees and composes a semantic vector along with these trees.

We organize our chapter as follows: Section 5.1 reviews related models, learning the composition function together with the sentence structure. Section 5.2 introduces our model, which is based on well-known components and could therefore accommodate a variety of parsing architectures such as graph parsers or attention matrices from Bert. In Section 5.3, we train and evaluate the full model with distant downstream supervision on textual entailment and semantic similarity tasks. Finally, in Section 5.4, we analyze how the initial parser supervision impacts the learned structures and the performance on downstream tasks.

5.1 Latent tree learning

Tree-structured models rely on an explicit and discrete structure to compute semantic representations. It favors the integration of linguistic information and inductive biases. Moreover, it favors compositional analysis since it explicitly relies on syntactic trees. However, such models require not only raw text but also linguistic structure in the form of parse trees to calculate the semantic representations. This prerequisite limits their use in practice because it requires annotations in the supervised case.

One method of overcoming this limitation is to induce trees from raw text and computes semantic representations along with the inferred structure. Such a method preserves explicit recursive computations and produces intelligible tree structures. Known as *latent tree learning*, these methods generally consist of two components: a parser and a composition function that uses the parses. The parser and composition function are learned jointly and are specific to a given task or domain.

The first set of latent tree models introduces an intermediate objective to train the parser component. Socher, Pennington, et al. (2011) parse the sentence by selecting and merging adjacent nodes. The parser model is trained using an auxiliary reconstruction task. The Shift reduce Parser-Interpreter Neural Network (SPINN) model from Bowman, Gauthier, et al. (2016) obtains the structure using a shift-reduce parser. The parser component also uses an intermediate objective that compares parses with gold-standard trees.

Maillard, S. Clark, and Yogatama (2019) explicitly compute a whole forest of potential binary parse trees for a sentence of N words. All possible partial trees of a sentence are stored in a chart data structure inspired by the CYK parser. The final tree is constructed as a soft combination of the constituents available in each chart cell, thus approximating discrete candidate selection and making the model entirely trainable using backpropagation. However, the linear increase in candidates with depth makes this algorithm memory intensive.

Yogatama et al. (2017) adapt the training of the SPINN model to make it fully differentiable. As such, the model does not require any structure supervision during training. Instead of

providing the model with the parse of the input, the procedure uses reinforcement learning (policy gradient methods) to discover the best tree structures for the task. However, the reinforcement learning strategy is notoriously slow and limits the convergence speed.

Choi, Yoo, and S. Lee (2018) propose a method that is both fully differentiable and maintains the discreteness of the parsing process. Contrary to Yogatama et al. (2017), it does not require the reinforcement learning artifice for training; contrary to Maillard, S. Clark, and Yogatama (2019), it computes a single discrete tree instead of combinations from partial trees. The methods proceed in $N - 1$ iterations to build a tree over N words. Using the Gumbel-Softmax estimator, two nodes are selected from the available candidates at each iteration. During the forward pass, the estimator is used as a discrete argmax function to select nodes to merge. During the backward pass, The estimator relaxes the discrete sampling operation so that it can be trained with backpropagation.

5.2 Unified parsing and compositional model

The earlier architectures listed above have practical limitations, requiring either a complex learning paradigm such as reinforcement learning, intensive computations, or requiring an external parser module. The method proposed in Choi, Yoo, and S. Lee (2018) overcomes these limits. However, Williams, Drozdov, and Bowman (2018) investigate the latent trees produced by Yogatama et al. (2017) and Choi, Yoo, and S. Lee (2018) and show neither method produces meaningful syntactic representations. The Gumbel-Softmax estimator outputs inconsistent trees across initializations, while reinforcement learning outputs trivial left-branching trees. Moreover, Choi, Yoo, and S. Lee (2018) produce trees by selecting and merging adjacent nodes. Therefore, it cannot directly use architectures designed for standard parsing formalisms such as dependency parsing algorithms.

In this section, we propose an original latent tree learning method. Besides addressing all the limitations listed above, our method relies on existing and well-known components. It is not limited to a particular parser architecture as long as it

is differentiable. Ultimately, our method offers the following benefits:

- infers an explicit tree structure and trains recursively a sentence embedding model within a unified architecture;
- provides end-to-end training by back-propagating the downstream task loss through the entire architecture;
- produces a discrete tree;
- accommodates any any graph-based dependency parser architecture.

Our model jointly performs sentence parsing and the prediction of a sentence embedding. The sentence embedding is predicted by a TreeLSTM whose tree structure is provided by a dependency parser.

Parsing model We use a standard dependency parsing structure, obtained using a graph-based biaffine dependency parser (Dozat and Manning 2017). Given an input sequence of n words, the parser computes a weighted matrix of size $n \times n$ for which each coordinate (i, j) is interpreted as a score for the ith word to be the head of the jth word. Given the un-normalized score matrix, the predicted tree is extracted using the MST algorithm. However, our model is agnostic to any graph-based parser architecture. This flexibility gives us the freedom to explore the impact of the parser choice (Section 5.4).

The procedure is formalized in two steps. First, in Eq. 5.2 to 5.4, it computes a weight matrix that is interpreted as weighted directed graph whose nodes are the sentence tokens:

$$\text{Biaff}(x_1, x_2) = x_1^T U x_2 + W^{(b)}(x_1 \oplus x_2) + b^{(b)} \tag{5.1}$$

$$a_k^{(dep)} = W^{(dep)} h_k + b^{(dep)} \tag{5.2}$$

$$a_j^{(head)} = W^{(head)} h_j + b^{(head)} \tag{5.3}$$

$$s_{kj}^{(arc)} = \text{Biaff}(a_k, a_j) \tag{5.4}$$

The second step performs parsing by computing a maximum spanning tree from the graph. As in Dozat and Manning (2017), we use the Max Spanning Tree (MST) algorithm (Chu 1965; Edmonds et al. 1967) to ensure the well-formedness of

the tree:

$$\alpha_{kj} = \mathbb{1}_{mst(s^{(arc)}_{kj})} s^{(arc)}_{kj} \tag{5.5}$$

Where α_{kj} is the probability of the edge linking node j to node k. For a given node k, there is at most one non-zero edge leading to its governor j.

Composition function Given a predicted tree structure, we recursively encode the sentence using a variant of the Child Sum Tree model from Tai, Socher, and Manning (2015) detailed below:

$$\tilde{h}_j = \sum_{k \in C(j)} \alpha_{kj} h_k, \tag{5.6}$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \tag{5.7}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \tag{5.8}$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \tag{5.9}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \tag{5.10}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{5.11}$$

$$h_j = o_j \odot \tanh(c_j), \tag{5.12}$$

Where in Eq. 5.6, $C(j)$ denotes the set of children of node j.

We use the embedding computed by the weighted TreeLSTM at the root of the tree as the sentence embedding. The tree shape and the edge weights are given by the best prediction of a graph parser. The equations from the TreeLSTM are the same than the one presented in Section 3.2.3, except for Eq. 5.6. **Crucially, in our case, Eq. 5.6 is a weighted sum rather than a standard sum and the weights are those α_{kj} provided by the parser.** The parsing model is linked to the TreeLSTM by the weights α_{kj}. This architecture allows us to update jointly the parser and the TreeLSTM weights using only the downstream task loss. The supervision comes only from the objective of the downstream task, and no intermediate structure target is required.

Our model, illustrated in Figure 5.2, is fully differentiable and preserves the discreteness of the tree composition process. It relies on a dependency parsing formalism and could accommodate any graph-based dependency parser. Intuitively, the model induces a direct link between the inference of the syntactic structure and the composition of the semantic representation. If a connection between two nodes i and j is irrelevant from a semantic standpoint, then its contribution α_{kj} into the construction of the hidden state $\tilde{h}_j$ (Eq. 5.6) is likely to be marginal. When training the model, if such connection becomes too tenuous, it becomes unlikely to be selected when selecting the maximum spanning tree from the graph in equation 5.5.

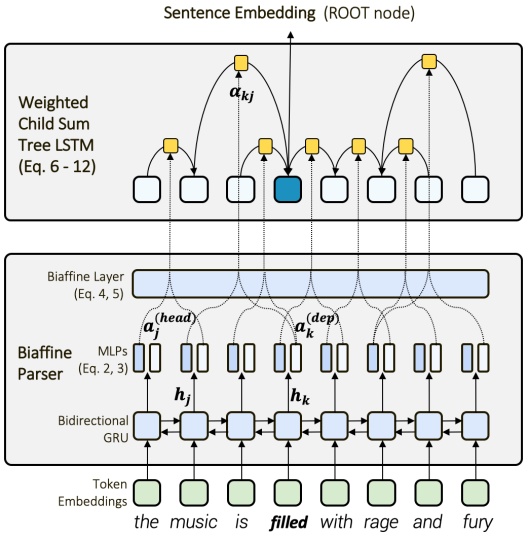


Figure 5.2: We illustrate the architecture detailed in Eq. 5.2 to 5.12. The Biaffine parser provides the sentence structure from which the TreeLSTM computes sentence embeddings. The full pipeline is differentiable as the TreeLSTM weights are given by the parser.

5.3 Evaluation on downstream tasks

Our architecture primarily aims to produce relevant embeddings for downstream tasks. To this end, we compare our setup with other models from the literature on various tasks. For this comparison, we first pre-train the parsing submodel on human-annotated sentences from the Penn Tree Bank (PTB) (Marcus, Santorini, and Marcinkiewicz 1993) converted to Stanford dependencies. We then fine-tune the parser's parameters on the task while training the full model.[1]

1: In this configuration, we observe pre-training the parser may cause weights α to become too large in Eq. 5.5. This leads to poor downstream performance. We correct this with a multiplicative parameter τ whose value is estimated during training. It means we replace Eq. 5.5 with: $\alpha_{kj} = \tau \cdot \mathbb{1}_{mst(s_{kj}^{(arc)})} s_{kj}^{(arc)}$ for the computation of tree weights.

5.3.1 Semantic textual similarity (STS)

We first evaluate our model on the SICK-R downstream task (Marelli et al. 2014), which is dedicated to assessing models' compositional properties. The dataset comprises 9,927 sentence pairs, distributed in a 4,500/500/4,927 train/dev/test split, annotated for semantic similarity on a 1 to 5 real range. A score of 5 means that the two sentences are completely equivalent and 1 that the two sentences are completely dissimilar. In between, their degree of equivalence differs given the proportion of shared topics and information between the two sentences. The dataset includes specific examples of variations on passive and active forms, quantifier and modifier switches, or negations. We extensively present the construction of the dataset in 7.2.2 and give some illustration examples of the task in Table 5.1.

Table 5.1: SICK-R is a Semantic Textual Similarity (STS) task for which labels are scores between 1 and 5. A score of 5 means that the two sentences are completely equivalent and 1 that the two sentences are completely dissimilar. In between, their degree of equivalence differs given the proportion of shared topics and information between the two sentences.

Sentence A	Sentence B	Target
"A man is singing a song and playing the guitar"	"A man is opening a package that contains headphones"	1.6
"Two dogs are playing by a tree"	"Two dogs are playing by a plant"	4.6
"A woman is riding a horse"	"A man is opening a small package that contains headphones"	1.0
"A potato is being sliced by a woman"	"A woman is slicing a carrot"	3.0
"A man is screaming"	"A man is scared"	3.6
"Men are sawing logs"	"Men are cutting wood"	4.5

Training configuration We use a similar training procedure as in Tai, Socher, and Manning (2015). We transform the target y from the SICK-R task into the distribution p defined by:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

We use a dedicated architecture to predict the similarity distribution from a pair of sentences. The similarity module takes as input a pair of sentence vectors h_L and h_L and computes their componentwise product $h_L \odot h_R$ and their absolute difference $|h_L - h_R|$. Given these features, we compute

the probability distribution $\hat{p}_\theta$ using a two-layer perceptron network (MLP):

$$\begin{aligned} h_\times &= h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\ h_s &= \sigma(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}), \\ \hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}), \end{aligned} \tag{5.13}$$

Where σ is the sigmoid function. We use the KL-divergence between the prediction $\hat{p}_\theta$ and the ground truth p as a/ training objective:

$$J(\theta) = \frac{1}{N} \sum_{k=1}^{N} \text{KL}(p^{(k)} || \hat{p}_\theta^{(k)}) + \lambda ||\theta||_2^2 \tag{5.14}$$

Finally during inference, the similarity score $\hat{y}$ is computed as $\hat{y} = r^\intercal \hat{p}_\theta$ with $r^\intercal = [1, \ldots, 5]$.

Hyper-parameters We set the hyperparameters in accordance with the choices made in Tai, Socher, and Manning (2015), such that we can directly compare our results in Table 5.2. For all experiments detailed in the current section, the batch size is fixed to 25, weight decay to $1e^{-4}$ and gradient clipping set to 5.0. The learning rate is set to 0.025 for the TreeLSTM parameters. When using a pre-training procedure for the parser, we set the learning rate to $5e^{-3}$ and use the following warm-up: for the first epoch, the parser is frozen. For the following epochs, all parameters are trained. At each epoch, the parser learning rate is divided by a factor of two. Without pre-training, the learning rate is set to $5e^{-4}$ for the parser. All model weights are initialized with a Xavier distribution. The hidden size of the similarity architecture is set to 50. The TreeLSTM hidden size is set to 150. We use the Adagrad optimizer. We do not apply any dropout. We perform training for a maximum of 20 epochs and stop when no improvement was observed on the development set for 3 consecutive epochs. Regarding the vocabulary, we limit the size to 20,000 words and initialize the embeddings layer with 300-dimensional GloVe embeddings.[2] The embeddings are not updated during training.[3]

Table 5.2 reports the results from the test set. As expected, structured models perform better than models using weaker underlying structures. We also observe that our model is competitive with a Bert-base upper-line. It is essential to note

2: https://nlp.stanford.edu/projects/glove/

3: We trained all models on a single 1080 Ti Nvidia GPU. Training time for each epoch is approximately 1 minute. The model counts 13.7M parameters. Data can be downloaded using the SentEval package https://github.com/facebookresearch/SentEval.

that BERT models are heavily pre-trained on vast corpora, whereas our structured models are trained only on the SICK-R and PTB data.

Table 5.2: Evaluation of the model on the SICK-R task: we pre-train our parsing module on the PTB and continue to update the full model on the SICK-R task. We compare with BERT and models relying on sequential and tree structures. We report Pearson correlation on the test set, by convention as $r \times 100$ (average and standard deviation from 5 runs). † indicates models that we trained.

Encoder	r
Bow†	78,2 (1,1)
LSTM†	84.6 (0.4)
Bidirectional LSTM†	85.1 (0.4)
N-ary TreeLSTM† (Tai, Socher, and Manning 2015)	85.3 (0.7)
Childsum TreeLSTM† (Tai, Socher, and Manning 2015)	86.5 (0.4)
BERT-base (Devlin et al. 2019)	87.3 (0.9)
Our model	
Unified TreeLSTM†	87.0 (0.3)

5.3.2 Textual entailment

We also test our model on the Stanford Natural Language Inference (SNLI) task (Bowman, Angeli, et al. 2015), which includes 570k pairs of sentences with the labels entailment, contradiction, and neutral. It is distributed in a 550k/10k/10k train/dev/test split. We already presented some examples from the SNLI task in Table 3.1. We reproduce some other examples in Table 5.3.

Table 5.3: SNLI examples presented in the original paper (Bowman, Angeli, et al. 2015) and extracted from the development section of the corpus.

Premise	Hypothesis	label
Two women are embracing while holding to go packages.	Two woman are holding packages.	entailment
Two men on bicycles competing in a race.	People are riding bikes.	entailment
Two women having drinks and smoking cigarettes at the bar.	Three women are at a bar.	contradiction
Two doctors perform surgery on patient.	Two doctors are performing surgery on a man.	neutral
A man in a black shirt is playing golf outside.	A man plays on a golf course to relax.	neutral

Training configuration We use a similar training procedure as in Choi, Yoo, and S. Lee (2018). A dedicated architecture is used to predict the similarity distribution from a pair of sentences. The similarity module takes as input a pair of sentence vectors h_L and h_L and computes their component-wise product $h_L \odot h_R$ and their absolute difference $|h_L - h_R|$.

Given these features, we compute the probability distribution $\hat{p}_\theta$ using a three-layer perceptron network (MLP):

$$\begin{aligned} h_\times &= h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\ h_s &= \text{ReLU}(W^{(1)}[h_\times, h_+, h_L, h_R] + b^{(1)}), \\ h_s &= \text{ReLU}(W^{(2)} h_s + b^{(2)}), \\ \hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}), \end{aligned} \tag{5.15}$$

We use the cross entropy loss between the prediction $\hat{p}_\theta$ and the ground truth p as a/ training objective:

$$J(\theta) = -\frac{1}{N} \sum_{k=1}^{N} p^{(k)} log \hat{p}_\theta^{(k)} + \lambda ||\theta||_2^2 \tag{5.16}$$

Hyper-parameters We set the hyperparameters in accordance with the choices made in Choi, Yoo, and S. Lee (2018), such that we can directly compare our results in Table 5.4. For all experiments detailed in Section 5.3.2, the batch size is fixed to 128, weight decay to 0, and gradient clipping set to 5.0. The learning rate is set to $1e^{-3}$ for the TreeLSTM and the parser. The hidden size of the similarity architecture is set to 1024. The TreeLSTM hidden size is set to 600. We use the Adam optimizer. We apply a 0.2 dropout within the similarity architecture. We perform training for a maximum of 20 epochs and stop when no improvement was observed on the development set for 3 consecutive epochs. Still following Choi, Yoo, and S. Lee (2018), we limit the size of vocabulary to 100,000 words and initialize the embeddings layer with 300-dimensional GloVe embeddings. The embeddings are not updated during training.

Encoder	**Test Acc.**
Spinn \w Reinforce (Yogatama et al. 2017)	80.5
CYK and TreeLSTM (Maillard, S. Clark, and Yogatama 2019)	81.6
Spinn (Bowman, Gauthier, et al. 2016)	83.2
ST-Gumbel (Choi, Yoo, and S. Lee 2018)	86.0
Structured Alignment (Yang Liu, Gardner, and Lapata 2018)	86.3
Bert-base (Z. Zhang et al. 2020)	90.7
Our model	
Unified TreeLSTM	85.0 (0.2)

Table 5.4: Evaluation of the model on the SNLI-R task: We pre-train our parsing module on the PTB and continue to update the full model on the SNLI task. We compare with Bert and latent tree learning models. We report the accuracy on the test set (average and standard deviation from 2 runs).

We report the results in Table 5.4. Our results are close to Choi, Yoo, and S. Lee (2018), which also compute semantic representations along with discrete tree structures but relies

on a distinct syntactic formalism. The performance gap can be attributed to the use of dependency instead of binary parsing. However, it is also important to note that we encode the leaf nodes using only static embeddings, while Choi, Yoo, and S. Lee (2018) apply sequential LSTMs to the leaf nodes, resulting in a hybrid model with dual latent structures. The authors affirm that "the LSTM applied to leaf nodes has a substantial gain over the basic leaf [affine transformation]". Based on their results, this transformation of the leaf node induces an accuracy improvement of about 1.4 points.

In models from Yang Liu, Gardner, and Lapata (2018) and Z. Zhang et al. (2020) sentences are encoded with direct interaction using an attention mechanism. These architectures relying on cross sentences attention outperform those without. We hypothesize that, on this textual entailment task, the final prediction cannot be directly deduced from both sentence embeddings. In this case, Bert and the structured alignment model have a clear advantage since they encode interactions between both sentences.

5.4 Impact of the parser initialization

Our framework primarily aims to be a structured sentence encoder. Accordingly, we have demonstrated in the previous section that our architecture is competitive with comparable approaches and might even be competitive with Bert-based models. We are also interested in interpreting the structures the model actually learns and how such structures impact downstream performance.

In the previous experimental section, we pre-trained the parser on human-annotated data. However, the optimal structure of a sentence may not derive from linguistic insights. It may also depend on computational factors. For example, the length of the the computational path from the root to the final representation could be an important factor. Tree-structured neural networks compute the root at the very last step, while in sequential LSTM, the computational path from the root to the final representation is longer. Finally, as explored in Chapter 4 some structures may be better adapted for a given task. For example, tree-structure may be more adapted for sentiment analysis but not be the best structure for a keyword extraction task.

In this section we perform an ablation study to better understand how the initialization of the parser impacts the resulting structures (Section 5.4.3) and the final downstream performance (Section 5.4.4). We begin by defining the different initialization scenarios we considered (Section 5.4.1 and Section 5.4.2). In all scenarios, we either continue to update the parser when fine-tuning the model on downstream tasks or freeze the parser and only train the TreeLSTM. These two configurations are indicated with ✓ and × symbols respectively.

5.4.1 Adjusting the proportion of linguistic annotations

Tree-structured models traditionally rely on linguistic structures obtained by parsers (Tai, Socher, and Manning 2015). Linguistic resources are available for languages such as English; it is technically possible to pre-train the parser. However, resources such as the PTB are not available in all languages. To better quantify the benefits of using linguistic annotations, we propose the following configurations, using various proportions of the PTB to initialize the parser:

- In the **PTB-All** configuration, the parser is previously pre-trained on the PTB. This configuration is the same as in Section 5.3.
- In the **PTB-∅** configuration, the parser parameters are randomly initialized
- We also consider an initialization with only a small proportion of the **PTB** and train a parser by only using 100 randomly selected samples. This configuration is referred as **PTB-100**.

5.4.2 Using unsupervised structures

We are also interested in structures emerging from large pre-trained models. Such models present similarities with recent graph parser architecture. Although they are not trained with a direct parsing objective, many lines of work investigate if attention matrices can reflect syntactic structures (K. Clark et al. 2019; Jawahar, Sagot, and Seddah 2019; Ravishankar et al. 2021) or, on the contrary, if it is efficient to integrate tree

structural information within transformers (J. Bai et al. 2021; Yau-Shian Wang, Hung-yi Lee, and Y. Chen 2019).

As stated in the Introduction, our model is agnostic to any graph-based dependency parser. It is therefore possible to use any model or heuristic to infer sentence structure. In particular, we can use a pre-trained model such as BERT to infer structures based on the internal representations it learns. We do not intend to provide an in-depth analysis of how BERT could be used for unsupervised parsing. Therefore, we do not extensively explore how BERT could accommodate parsing tasks. However, we instead propose a proof-of-concept that our model can accommodate a large variety of graph-based parsers and show it is indeed possible to use BERT as an unsupervised parser in our case.

BERT relies upon the self-attention mechanism. Inside each layer, tokens are computed as a weighted combination of each other. For each token x, a query and key vector are computed using a linear transformation detailed in Eq 5.17. Given these vector tuples, the attention weights s are computed following Eq 5.18 in which N refers to the dimension of the query and key vectors.

$$q_j, k_j = W^{(q,k)} x_j + b^{(q,k)} \tag{5.17}$$

$$s_{kj} = \text{softmax}\left(\frac{k_k \cdot q_j}{\sqrt{N}}\right) \tag{5.18}$$

We induce a tree structure following a procedure close to the one used in Ravishankar et al. (2021).[4] The method interprets the combination weights s as a weighted graph whose nodes are tokens. We then apply Eq 5.4 to induce a maximum spanning tree from the attention matrix as detailed in Section 5.2. We make use of the last layer and induce a tree from the first attention head.[5] Given the tree structure induced from BERT, we apply our TREELSTM model detailed in Eq. 5.6 to 5.12. We stress the fact that in this configuration, we only use BERT as an unsupervised parser to infer a sentence structure. The semantic composition along with the structure to produce a sentence embedding is solely computed by the weighted TREELSTM.

4: Ravishankar et al. (2021) decode dependency trees from attention matrices using the Chu-LiuEdmonds maximum spanning tree algorithm (Edmonds et al. 1967) and compare them with gold treebank using the Undirected Unlabeled Attachment Score (UUAS)—the percentage of undirected edges recovered correctly.

5: As mentionned earlier, we only aim at proposing a proof-of-concept here. Therefore, we do not test all possible heads and layers to induce trees.

5.4.3 Impact of the initialization on parses

In this section, we analyze to which extent the structures generated by our model are comparable with meaningful linguistic annotations. We compare the parses generated by two distinct models differing by their initialization on the development set of both tasks. Our reference is the silver parses from the PTB-All configuration, where the parser is previously pre-trained on the full PTB and not updated during training.

In Table 5.5, we measure the Unlabeled Attachment Score (UAS) between the two parsers, that is, the ratio from the number of common arcs between two parses by the total number of arcs.

Parser 1	Parser 2	SICK-R (dev UAS)	SNLI (dev UAS)
Impact of parser fine-tuning			
PTB-100 (✓)	PTB-100 (×)	85.2 (1.5)	5.6 (1.9)
PTB-All (✓)	PTB-All (×)	98.4 (0.1)	11.7 (0.9)
Impact of the PTB sample size			
PTB-100 (✓)	PTB-∅ (✓)	6.3 (0.0)	10.1 (10.7)
PTB-All (✓)	PTB-∅ (✓)	10.1 (0.0)	15.1 (15.4)
PTB-All (✓)	PTB-100 (✓)	76.9 (0.7)	0.3 (0.2)
Unsupervised parser			
BERT (×)	PTB-All (×)	—	13.0 (4.9)
BERT (✓)	PTB-All (×)	—	13.7 (2.7)

Table 5.5: Impact of the parser initialization on parses: we compare the parses from the SICK-R and SNLI development sets using different parser initializations. We obtained the PTB parses with the graph parser initialized on a given proportion of the PTB (Section 5.2). Regarding BERT , we inferred the structures from the pattern learn by the pre-trained model (Section 5.4). We either continue to update the parser (✓) when fine-tuning the model on downstream tasks or freeze the parser (×) and only train the TREELSTM. UAS corresponds to the mean pairwise comparison of two configurations between two runs (standard deviation in parentheses).

We observe distinct behaviors given both tasks. We believe this effect is due to the differences between training configurations—detailed in Section 5.3.2 and 5.3.1. In particular, we use the Adagrad optimizer for the SICK-R task and Adam for the SNLI task.

For the SICK-R task, the UAS between PTB-∅ and PTB-All are very low. This reveals that the parses obtained with only downstream task supervision overlap very little with with gold linguistic parses. In this regard, we share the observation from Williams, Drozdov, and Bowman (2018) that latent trees obtained from sole downstream supervision

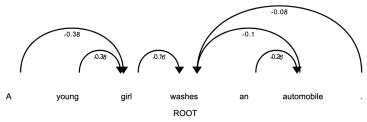
are not meaningful in syntax. However, PTB-All and PTB-100 are remarkably close; only a few PTB samples are needed to obtain intelligible linguistic parses with our setup. Regarding the PTB-100 configuration, we note an evolution of the parses when fine-tuning on the downstream task. We hypothesize that the model can adapt itself to the dataset's specificity.

Regarding the SNLI task, fine-tuning the parser deeply impacts the shape of the parses. Depending from the initialization, parses will converge to distinct structures. Indeed, the UAS between all configurations is very low. Moreover, we observe that when using a random initialization (PTB-$\emptyset$), the standard deviation between the UAS from various runs is very high. This reveals that without fixed initialization, the parses tend to show some instability.

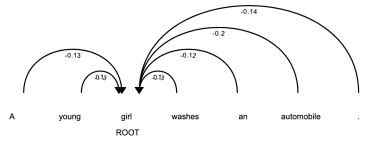
For the initialization with an unsupervised structure, we only evaluate our setup on the SNLI task, which has more training samples. We compare the structures obtained with Bert with the silver trees from the PTB-All-$\times$ configuration. We present the mean UAS over the trees obtained for all attention heads. The standard deviation is relatively high, pointing underlying structures differ given the attention head. Nonetheless, self-supervised structures do not align well with linguistic insights. When updating Bert together with the TreeLSTM, the UAS increases while the standard deviation decreases. As Bert is fine-tuned, structures tend to become more standard and present slightly more similarities with linguistic patterns.

Visualization of the parses We illustrate the effect summarized in Table 5.5 on some chosen examples. Figures from the first column (5.3a, 5.3c and 5.3e) show the parses obtained without updating the parser component on the downstream task. Figures from the second column (5.3b, 5.3d and 5.3f) show the evolution of the parses for the same initialization but after fine-tuning the parser on the SNLI task. Figures from the first raw (5.3a and 5.3b) are initialized using the full PTB, the second raw (5.3c and 5.3d) is initialized using 100 PTB samples, while the one from the last raw (5.3e and 5.3f) are initialized using unsupervised patterns.

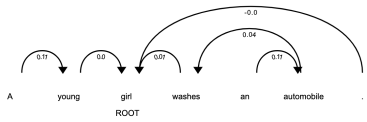
As a result of the fine-tuning, we observe that trees evolve into trivial structures and tend to connect every node to an arbitrary root. We postulate that such trivial structures



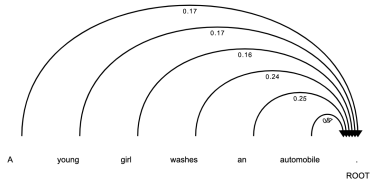
(a) Parse obtained using the the PTB-All (×) configuration.



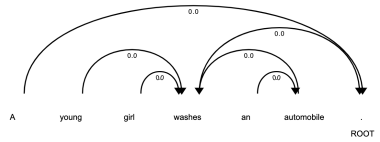
(b) Parse obtained using the the PTB-All (✓) configuration.



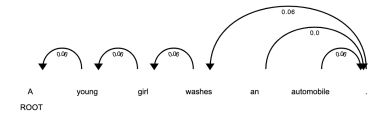
(c) Parse obtained using the the PTB-100 (×) configuration.



(d) Parse obtained using the the PTB-100 (✓) configuration.



(e) Parse obtained using the attention head #1 and without updating Bert.



(f) Parse obtained using the attention head #1 and updating Bert.

Figure 5.3: Example of parse obtained using various configurations from our model. The parser component is initialized on PTB-All (5.3a, 5.3b), PTB-100 (5.3c, 5.3d) or Bert (5.3e, 5.3f). We either freeze (×) or update (✓) the parser during the fine tuning on the SNLI. We include the weights α produced from the parser. We report the accuracy from a single run on the test set.

present advantages from a computational standpoint. Shi et al. (2018) also observe that trivial trees without syntax yield better results than syntax and latent trees. They postulate that balanced binary trees benefit from two advantages. First, balanced trees treat all leaf nodes equally, making it easier to select essential information from all words within a sentence automatically. Second, balanced trees have shorter tree depths, which induces a shorter path for propagating information from leaves to roots, thereby reducing propagation errors.

For Bert parser initialization, we observe the fine-tuning produces rather sequential patterns, with words connected to direct neighbors. Some isolated groups of words also present inner connections.

5.4.4 Impact of the initialization on downstream tasks

We observed in previous Section 5.4.3 that the initialization and the training configuration of the parser component

deeply impact the resulting parses. We now study the impact of the parser initialization on downstream performance.

Table 5.6: Impact of the parser initialization on downstream task performance: We pre-train the parser module with a given sample size from the PTB. We either freeze (×) or update (✓) the parser during the fine-tuning. We report the average score test set from 5 runs for SICK-R and 2 runs for SNLI (the score from the development set are in parentheses). We report Pearson correlation by convention as $r \times 100$.

PTB sample size	Parser fine-tuning	SICK-R (r)	SNLI (Acc.)
Linguistic annotations			
PTB-∅	✓	85.6 (85.6)	84.6 (85.5)
PTB-100	×	86.4 (86.6)	84.5 (85.5)
PTB-100	✓	86.5 (86.9)	84.9 (85.8)
PTB-All	×	86.8 (87.2)	85.0 (85.8)
PTB-All	✓	87.0 (87.5)	85.0 (85.5)
Unsupervised parser			
BERT	×	—	84.4 (85.3)
BERT	✓	—	84.6 (85.1)

In Table 5.6, we compare the impact of the different initializations for both tasks. For each setup, we report the Pearson correlation on the test set of the SICK-R task and the accuracy on the test set from the SNLI task.

We either freeze the parser component or continue to update it, given the downstream loss for each initialization. Fine-tuning the parser on the task generally leads to an improvement in the downstream results. In that regard, we share the observation from other latent tree learning methods (Choi, Yoo, and S. Lee 2018; Maillard, S. Clark, and Yogatama 2019); models jointly learning the parsing and composition function outperform those with a fixed structure.

We also observe that models using the full or partial annotated data outperform models relying on the sole downstream supervision (PTB-∅). This observation is more clear on the SICK-R task. We previously observed that fine-tuning the parser can lead to tree structure diverging from linguistic patterns. Nonetheless, human annotations appear to be a good initialization for our model regarding the downstream performance.

We can observe that models relying on linguistic-driven structures achieve better performance. Nonetheless, the difference is thin, and we present an average score across trees obtained from all attention heads. Therefore some attention heads might present structures as efficient as linguistic patterns.

5.5 Conclusion and future work

We evaluate our model on textual entailment and semantic similarity tasks. Regarding the textual similarity task, we show that our setup is competitive with BERT base, although the latest is trained on datasets many orders of magnitude larger. We explore to which extent the trees produced by our model compare with linguistic structures and how this initialization impacts downstream performance. We empirically observe that downstream supervision troubles producing stable parses and preserving linguistically relevant structures. To encourage convergence towards readable linguistic structures, we examine a number of initialization setups. Depending on the optimization setup, the parse tree may present instability. We also observe that our structures often converge toward trivial branching patterns, which have little in common with gold linguistic parses. However, with respect to the downstream performance, linguistic insights appear to be a relevant initialization.

A study of the shallow structure in transformer models

6

> *At midnight in the month of June,*
> *I stand beneath the mystic moon.*
> *An opiate vapour, dewy, dim,*
> *Exhales from out her golden rim,*
> *And, softly dripping, drop by drop,*
> *Upon the quiet mountain top.*
> *Steals drowsily and musically*
> *Into the universal valley.*
> [...]
>
> — **Edgar Allan Poe**
> The Sleeper, 1831

The previous chapters focused on tree-structured models. We observed that such architectures have practical limitations, including low computational efficiency due to batching difficulties and the frequent requirement to infer sample structure from annotated data. Because of their practical constraints and limited performance increment, the community puts the focus on alternative methods such as recurrent neural networks (Cho et al. 2014; Hochreiter and Schmidhuber 1997) or transformers (Vaswani et al. 2017) that have gained an increasing popularity in recent years. Unlike tree-based models, these methods do not require costly annotations.

Transformers introduce a profound paradigm shift with recurrent and recursive architectures. For the latter, the input structure determines the computational path: recurrent networks process words sequentially given their order; recursive networks process words in a bottom-up manner—starting from the leaves up to the root. In contrast, Transformers process all words simultaneously through a fixed number of layers and do not appear to enforce an obvious structure. However, as many results suggest, these new models acquire some sort of structure. In particular, Linzen, Dupoux, and Goldberg (2016) probe LSTM architecture for grammar competencies such as subject-verb agreement. While LSTMs do not have inherent representations of hierarchical structure, they can capture a substantial amount of grammatical structure when explicitly supervised. Jawahar, Sagot, and Seddah (2019) perform a series of experiments to investigate the nature of the representations learned by different layers

of Bert. They observe that the nature of information encoded by Bert evolves with the layer depth: beginning with surface features in the earlier layers and progressing to syntactic and semantic aspects at the top. Based on their findings, Bert representations reflect linguistic information in a compositional manner that mimics classical tree-like structures. K. Clark et al. (2019) analyze the patterns across Bert 's attention heads and probe them for linguistic phenomena. They demonstrate that certain attention heads correlate well with linguistic notions of syntax and coreference.

This chapter interprets transformers as graph neural networks. We also extend this formalism beyond transformers to sequential and tree-structured models. We use it as a new analysis grid for these architectures. While most recent model analysis work for transformers focuses on probing vector representations or attention maps, our formalism provides a new interpretation path for such architectures. Given our interpretation, we conjecture that layers do not act as different feature extractors—each specialized at a given abstraction level. But, that the number of layer applications gradually abstracts the surface information into semantic knowledge. Layers are thus part of an iterative process where the token contextualized representations are progressively refined. We can thus study how transformers process text in the light of this iterative transformation.

We organize our argumentation as follows: Section 6.1 interprets transformers as structured neural networks and layers as operations on fully connected graphs. We then challenge our formalism by conducting an empirical investigation of the role of multiple layers in deep transformer models. Section 6.2 proposes a variant of Albert (Simoulin and Crabbé 2021b) that dynamically adapts the number of layers applied to each token. In particular, we encourage our model to be parsimonious and limit the total number of iterations performed on each token. In Section 6.3, we analyze token transformation across the network depth and during the pre-training (Section 6.3.1), fine-tuning and inference (Section 6.3.2).

6.1 Transformer as graph neural networks

In this section, we begin by briefly introducing graph neural networks (GNN) and reviewing a few key concepts. Next, we formalize transformers as GNNs and discuss how this interpretation offers new analysis methods or reflection for architecture evolutions.

6.1.1 Defining graph neural networks

In this section, we summarize the graph neural network framework as defined in Hamilton (2020). Graph neural networks operate on graph-structured data. We define a graph $\mathcal{G}$ by a tuple of sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. With $\mathcal{V}$ the set of vertices and $\mathcal{E}$ the set of edges between the node. A graph is a ubiquitous data structure that can describe many complex systems. For example, molecules are a group of atoms held together by chemical bonds. Chemical graphs are full-fledged representations, with vertices corresponding to atoms and edges to chemical bonds.

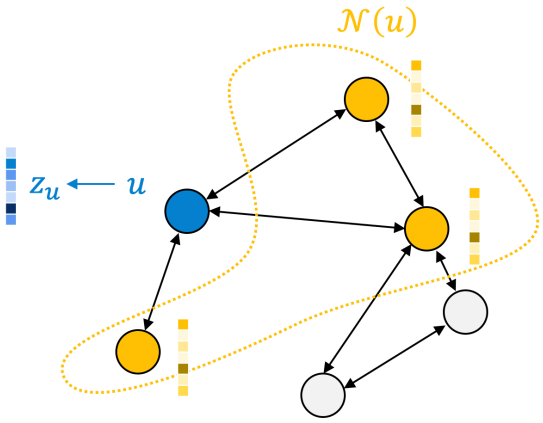


Figure 6.1: Illustration of a graph neural network (GNN) applied to an instance of a graph. The graph neural network takes the graph along with a set of node features (represented as vectors on the figure) as input and associates each node $u \in \mathcal{V}$ from the graph to an embedding z_u. For the node u (in blue), we illustrate its neighborhood $\mathcal{N}(u)$ as all the nodes directly connected to u (in yellow).

Graph neural networks take as input a graph $\mathcal{G}$ along with a set of node features $X \in \mathbb{R}^{d \times |\mathcal{V}|}$, with d the network hidden size. We illustrate a graph neural network in Figure 6.1. Graph neural networks generate a node embedding $z_u, \forall u \in \mathcal{V}$. They iteratively update node hidden embeddings $h_u^{(k)}$ using a neural message passing process. We can decompose each iteration into two steps: First, an *aggregation* step (Equation 6.1),

that aggregate the information from all nodes directly connected to u, that we define as the graph neighborhood $\mathcal{N}(u)$. Then an *update* step (Equation 6.2) that update each hidden embedding given its neighborhood aggregated information and its previous state. We illustrate the iterative process in Figure 6.2.

$$m_{\mathcal{N}(u)}^{(k)} = \text{aggregate}^{(k)}\left(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right), \tag{6.1}$$

$$h_u^{(k+1)} = \text{update}^{(k)}\left(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}\right), \tag{6.2}$$

1: The aggregate function takes a set as input and is therefore permutation invariant by design.

Here, the aggregate and update function are differentiable and $m_{\mathcal{N}(u)}$ is the *message* aggregated from u neighborhood. [1]

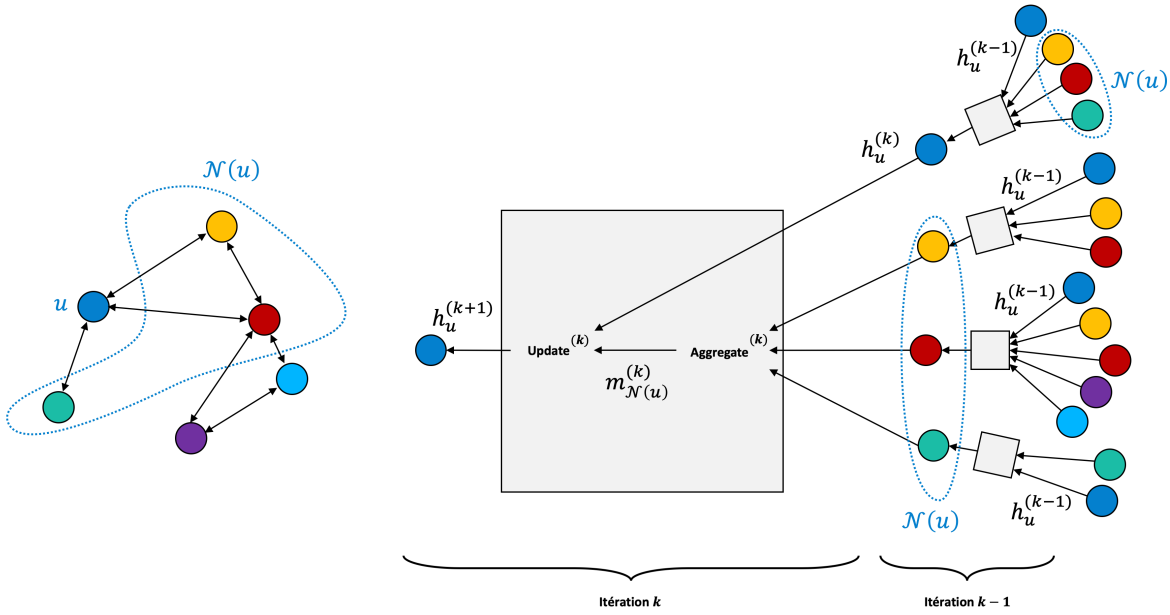


Figure 6.2: Illustration of one iteration to compute the hidden state of the node u (in dark blue). We decompose the iteration into two steps: the aggregation (Equation 6.1) and update (Equation 6.2) step. We illustrate two iterations, at step k to compute h_u^{k+1} and $k-1$ to compute h_u^k. We adapted the figure from Hamilton (2020).

For simplification, we can add self-loops to the input graph such that u is included in its own neighborhood $\mathcal{N}(u)$. The message passing iteration can then be defined using a single equation and we implicitly define the *update* step in the *aggregate* step (Equation 6.3). We illustrate the iterative process with this simplification in Figure 6.3.

$$h_u^{(k+1)} = \text{aggregate}^{(k)}\left(\{h_v^{(k)}, \forall v \in \mathcal{N}(u) \cup \{u\}\}\right), \quad (6.3)$$

Equation 6.4 defines the embedding of each node as its hidden state after K message passing iterations:

$$z_u = h_u^{(K)}, \forall u \in \mathcal{V}, \quad (6.4)$$

At each iteration, each node aggregates information from its k-hop neighbors. Node embeddings therefore encode structural and feature-based information.

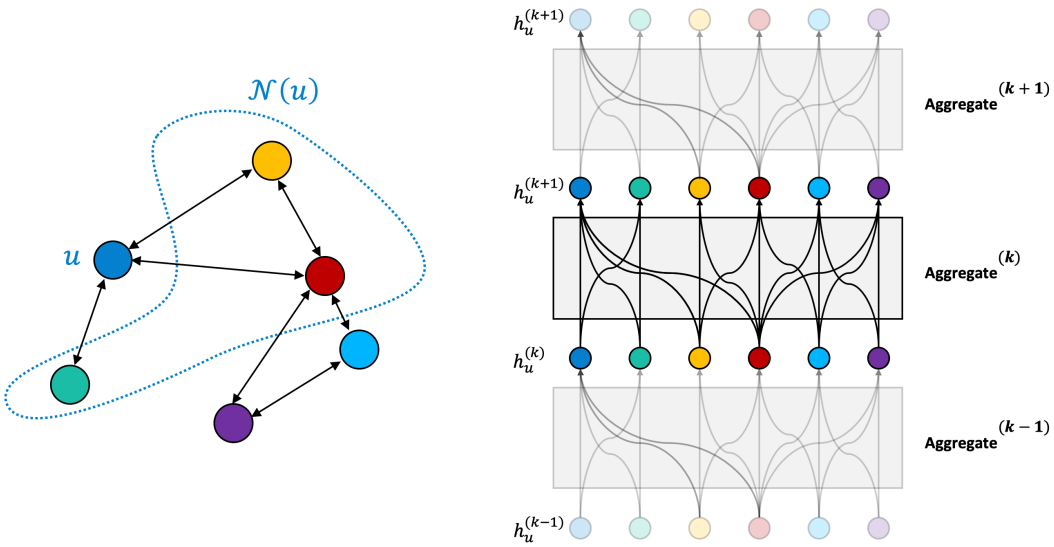


Figure 6.3: Illustration of one iteration to compute the nodes hidden state. We include each nodes in its own neighborhood $\mathcal{N}(u)$ and thus implicitly define the *update* step in the *aggregate* step (Equation 6.3). We illustrate iterations, at step k to $k+1$.

6.1.2 Defining transformer's message passing functions

Transformers (Vaswani et al. 2017) may easily be defined under the graph neural network framework. They transform inputs through a fixed number of layers. As detailed in Section 3.2.4, each layer is composed of a multi-head attention (MHA) block and a feed-forward network (FFN). The self-attention mechanism that transforms a *set of vectors* into what

is known as *contextualized vectors*. Each contextualized vector is a weighted average of the vectors from the original set. Since attention composes every vector from the set, we can consider that transformers operate on fully connected graphs. We can adapt the Equation 6.3 for transformers and compute the contextualized representation h_u of given token u in an input text as follow:[2]

2: In fact, we can also formally decompose transformer layers by considering the MHA layer as the aggregate function and the FFN layer as the update function. This interpretation may provide a formal background to analyze the role of the FFN.

$$h_u^0 = W^{(e)}u + W^{(p)} \tag{6.5}$$

$$h_u^{(k+1)} = \text{FFN}^{(k)}\left(\text{MHA}^{(k)}\left(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\right) + h_v^{(k)}\right), \tag{6.6}$$

Where Equation 6.6 is the first layer that encodes words using $W^{(e)}$ embedding layer summed with positional embeddings layer $W^{(p)}$. The neighborhood $\mathcal{N}(u)$ of each token u, corresponds to every token in the sentence, including the token u itself.

6.1.3 Tricks and limits

We can interpret transformers as GNNs operating on a fully connected graph. However, in the specific case of NLP, transformers should preserve the information about the sequential order of words. In that regard, transformers borrow common tricks and mechanisms from graph neural networks.

The transformer formalization encodes the linear order of words in the positional embedding layer $W^{(p)}$ from Equation 6.5. We can interpret the non-contextualized embeddings $h_u^0, \forall u \in \mathcal{V}$ as the set of initial node features $X \in \mathbb{R}^{d \times |\mathcal{V}|}$ defined in Section 6.1.1. As a result, the message passing iteration defined in Equation 6.6 is permutation invariant, a standard property in traditional graph neural networks.

It is also essential to preserve the word information across the iterations such that $h_u^{(k)}$ indeed captures the contextualized information about the token u. This issue of preserving token identities is addressed using the skip connection: before and after the FFN layers, we add the source to the current hidden state. The same mechanism is also commonly used in GNNs to avoid over-smoothing. As detailed in Hamilton (2020), this issue happens when node-specific information

is lost after several message passing iterations. In such a case, the representation of every node tends to become very similar. Commonly this issue is addressed by including the previous hidden state together with the aggregated state in the computation of the updated state. This skip connection aims at explicitly preserving information from the previous iteration during the update.

6.1.4 Interpretation

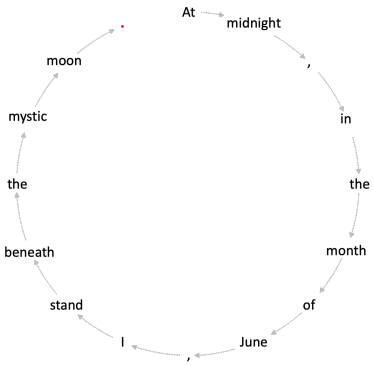
The formalization of transformers as graph neural networks opens an original angle to interpret them. The mechanism of transformer layers is often compared to intuitive NLP pipelines (Tenney, D. Das, and Pavlick 2019). Starting with the lower layers encoding surface information, middle layers encoding syntax, and higher layers encoding semantics (Jawahar, Sagot, and Seddah 2019; Peters et al. 2018).

In the graph neural network perspective, we conjecture that transformers progressively refine the feature through an iterative message passing process. As described in Xin et al. (2020). become more fine-grained at each iteration. This also provide a new interpretation path for Albert (Lan et al. 2020). The model is based on the transformer architecture, except that weights are tied across layers. In out GNN interpretation, the model uses the same message passing function at each iteration such that, in Equation 6.6, the functions $FFN^{(k)}$ and $MHA^{(k)}$ are the same for each iteration k.

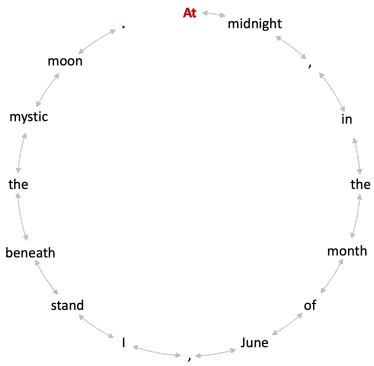
6.1.5 Graphs, trees, sequences

It is also possible to extend the graph neural network formalism for other NLP models. Tree-structured models operate on trees, which are directed acyclic graphs. The Tree-LSTM equations are detailed in Section 3.2.3. They may also be formatted in aggregate and update functions. In this case, the aggregate function consists of the simple sum of the node neighborhood hidden states. A key aspect is that the graph defined by Tree-LSTM does not contain any loop. the message passes in a bottom-up manner, starting from the leaf to the root. Since they have no dependant, the leaf nodes computation will be the same after the first iteration such that $\forall k \geq 2, h_u^{(k)} = h_u^{(k+1)}$. Similarly, for the computation of

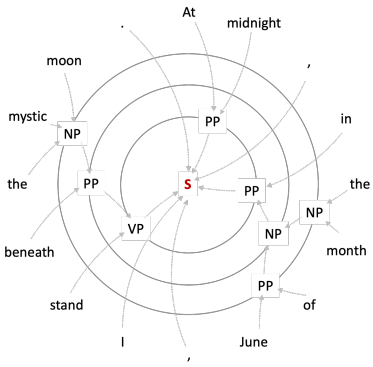
the leaf node parents, the inputs will be the same starting from the second iteration. After a number of iterations equal to the tree-depth, the value of all node hidden states will be determined.



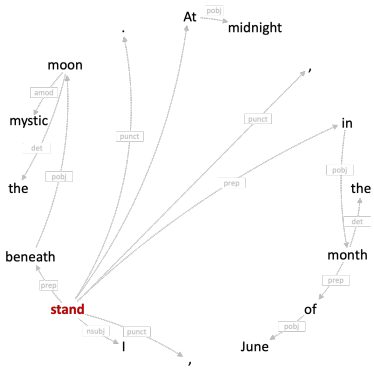
(a) Unidirectional RNN



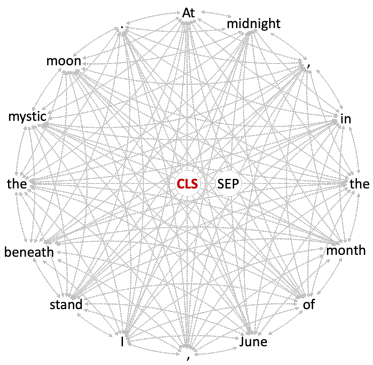
(b) Bidirectional RNN



(c) N-ary Tree RNN



(d) Childsum Tree RNN



(e) Transformer NN

Figure 6.4: Latent structure for distinct standard NLP architectures. Illustration on the first line of the poem: "The Sleeper" by Edgar Allan Poe (published 1831). We obtain the constituency tree for Figure 6.4c using the Berkeley neural parser (Kitaev and Klein 2018). For Figure 6.4d, we parse the sentence using the dependency parser from spaCy (`https://spacy.io/api/dependencyparser`) (Honnibal and Johnson 2015).

Sequential LSTM may also be considered a particular kind of graph neural network. A sequence is indeed equivalent to an unary directed acyclic graph. As for the Tree-LSTM, the LSTM equations may also be separated in a aggregate and update functions. In this case, the aggregate function is just the identity function since each node has exactly one child.

As for the Tree-LSTM, after a number of iterations equal to the sequence length, the value of all node hidden states will be determined.

We illustrate the input structure corresponding to the different architectures in Figure 6.4. Sequential RNN or TreeRNN operate on sparse graph. Node and edge numbers are equivalent in order of magnitude: $|\mathcal{V}| \approx |\mathcal{E}|$. On the contrary, transformer operate on a fully connected graph. The number of edges equals the number of nodes squared: $|\mathcal{E}| = |\mathcal{V}|^2$.

6.1.6 Are transformers over-parametrized?

It looks like the answer is included in the question. Transformers are indeed admittedly over-parametrized in the literature (D. Chen et al. 2020; Hou et al. 2020; Voita et al. 2019). However, the role of this over-parametrization is not well understood. Transformer layersliterature are suspected to be highly redundant (W. Liu et al. 2020) and to cause over-fitting (Fan, Grave, and Joulin 2020; W. Zhou et al. 2020).

In the light of our GNN analysis transformers should naturally share parameters across layers. As mentioned earlier, the Albert model (Lan et al. 2020) already implements this key specificity by tying weights across layers. Similarly, Tree-LSTMs or sequential LSTMs, which can be interpreted as GNN other specific graphs, also share their parameters across iterations.

However, the critical distinction between transformers and trees and sequences recurrent neural networks is the properties of the latent graph structure. Trees and sequences will converge after a finite number of message passing iterations since they do not include any loop within their structure. However, transformers operate on a fully connected graph. tokens' hidden states cannot be computed in a given hierarchical order since they all depend on each other.

We hypothesize that tokens' hidden states will eventually converge toward a fixed point.[3] We also hypothesize that some tokens require more iterations than others and that this convergence process will depend on the word and its context. Indeed, many studies show that the weighted graph formed by attention weights is not homogeneous. Some tokens contribute in large proportion to the update of another

3: Concerning this, J. Zhou et al. (2020) states that such fixed point is uniquely defined with the assumption that aggregate is a contraction map in Equation 6.3. By definition a contraction map is a function such that there is some non negative real number $0 \leq k < 1$ such that for all vector x and y, $d(f(x), f(y)) \leq k\, d(x, y)$, with d a distance metric over our vector space.

token, while some have a marginal or null contribution to the attention weights.

6.2 Dynamic transformer depth

We proposed to formalize transformers as graph neural networks. In light of this interpretation, we justified the possibility of sharing parameters across layers. We also formulated two hypothesis:

- Tokens' hidden states will eventually converge towards a fixed point;
- Some nodes require more iterations than others, and this convergence process will depend on the token itself and its context.

This section aims to provide empirical evidence to support this hypothesis. We design a variant of Albert that dynamically adapts the number of layers for each token of the input. We analyze the distribution of these iterations during pre-training, fine-tuning, and inference. We organize the section as follows: after reviewing the related work (Section 6.2.1), we detail the model (Section 6.2.2) and the training methodology (Section 6.2.4 and Section 6.2.3). In particular, we encourage our model to be parsimonious and limit the total number of iterations performed on each token. In Section 6.3, we analyze iterations of the model during pre-training, fine-tuning, and inference.

6.2.1 Related work

Adapting the transformer depth is an active subject of research. In particular, deep transformer models are suspected of struggling to adapt to different difficulty levels. While large models correctly predict difficult examples, they over-calculate simpler inputs (W. Liu et al. 2020). This issue can be addressed using *early-exit*: some samples might be sufficiently simple to classify using intermediate features.

Some models add a classifier to each layer (W. Liu et al. 2020; Xin et al. 2020; W. Zhou et al. 2020). After each layer, given the classifier output, the model either immediately returns the output or passes the sample to the next layer. Exiting

too late may even have negative impacts due to the network "over-thinking" the input (Kaya, Hong, and Dumitras 2019).

Ongoing research also refines the application of layers at the token level. B. Wang and Kuo (2020) build sentence embeddings by combining token representations from distinct layers. Elbayad et al. (2020) and Dehghani et al. (2019) successfully use dynamic layers depth at the token level for full transformers (encoder-decoder). However, to the best of our knowledge, our attempt is the first to apply such a mechanism to encoder only transformers and to provide an analysis of the process.

6.2.2 Model architecture

In this section, we detail the model architecture, illustrated in Figure 6.5, and pre-training procedure.

We use a multi-layer transformer encoder (Devlin et al. 2019) which transforms a context vector of tokens $(u_1 \cdots u_T)$ through a stack of L transformer encoder layers (Eq. 6.7, 6.8). We use weight tying across layers and apply the same transformation function at each iteration (Lan et al. 2020).

$$h_t^0 = W^{(e)} u_t + W^{(p)} \tag{6.7}$$

$$h_t^n = \mathsf{layer}(h_t^{n-1}) \quad \forall n \in [1, L] \tag{6.8}$$

For the first layer, $W^{(e)}$ is the token embedding matrix, and $W^{(p)}$ the position embedding matrix.

We augment the model with a halting mechanism, which allows the model to dynamically adjust the number of iterations for each token (Eq. 6.9 to 6.14). We directly adapted this mechanism from Graves (2016). The main distinction with the original version is using a transformer model instead of a recurrent state transition model. The mechanism works as follows: at each iteration n, we add the following operations after Eq. 6.8. We assign a probability to stop p_t^n for each token at index t (Eq. 6.9).

Given this probability, we compute an updated weight λ_t^n (Eq. 6.10), which we use to compute the final state as the

linear convex combination between the previous and current hidden state (Eq. 6.11).

$$p_t^n = \sigma\left(W_h h_t^n + b_n\right) \tag{6.9}$$
$$\lambda_t^n = p_t^n \text{ if } n < N_t, R_t \text{ elif } n = N_t, \text{ else } 0 \tag{6.10}$$
$$h_t^n = \lambda_t^n h_t^n + (1 - \lambda_t^n) h_t^{n-1} \tag{6.11}$$

With σ the sigmoid function. We define the remainder R_t and the number of iterations for the token at index t, N_t with:

$$R_t = 1 - \sum_{l=1}^{N_t - 1} p_t^l. \quad N_t = \min_{n'} \sum_{n=1}^{n'} p_t^n \geq 1 - \epsilon \tag{6.12}$$

As soon as the sum of the probability becomes greater than 1, the update weights λ_t^n are set to 0, and the token is not updated anymore (Eq. 6.10). A small ϵ factor ensures that the network can stop after the first iteration (Eq. 6.12).

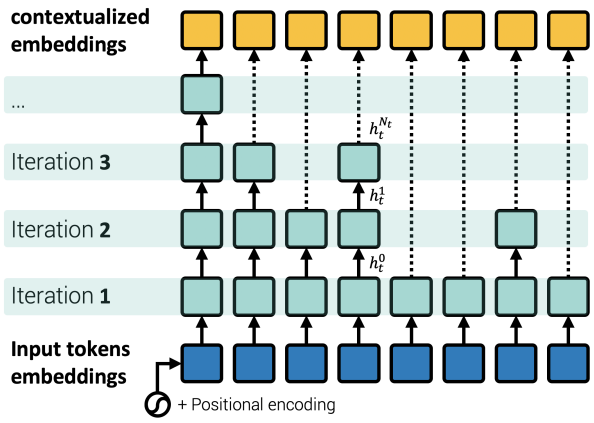


Figure 6.5: As in the Albert model, tokens are transformed through the iterative application of a transformer encoder layer. Our model's key specificity is the application of the halting mechanism, which dynamically adjusts the number of iterations for each token.

6.2.3 Pre-training objective

During the pre-training phase, we train the model with the sentence order prediction (*sop*) — the task introduced in Lan et al. (2020) that classifies whether segments from the input sequence follow the original order or were swapped — and the masked language model task (*mlm*) (Devlin et al. 2019). We also encourage the network to minimize the number of iterations by directly adding the ponder cost into Albert

pre-training objective. Given a length T input sequence $\mathbf{u}$, Graves (2016) defines the *ponder cost* $\mathcal{P}(\mathbf{u})$ as:

$$\mathcal{P}(\mathbf{u}) = \sum_{t=1}^{T} N_t + R_t \tag{6.13}$$

We define the final pre-training loss as the following sum:

$$\hat{\mathcal{L}} = \mathcal{L}_{sop} + \mathcal{L}_{mlm} + \tau\mathcal{P} \tag{6.14}$$

where τ is a *time penalty* parameter that weights the relative cost of computation versus error.

6.2.4 Datum and infrastructure

We follow the protocol from Albert and pre-train the model with BookCorpus (Zhu et al. 2015) and English Wikipedia. We reduce the maximum input length to 128 and the number of training steps to 112,500.[4] We use a lowercase vocabulary of size 30,000 tokenized using SentencePiece. We train all our models on a single TPU v2-8 from Google Colab Pro[5] and accumulate gradients to preserve a 4,096 batch size. We optimize the parameters using Lamb with a learning rate at 1.76e-3.

4: As emphasized in https://github.com/google-research/bert, longer sequences are computationally expensive. To lighten the pre-training process, they advise using 128 sentence length and increasing the length to 512 only for the last 10% of the training to train the positional embeddings. In this work, we only perform the first 90% steps as we are not looking for brute force performance.

5: https://colab.research.google.com/

6.3 Experiments

We now analyze our iterative model properties during pre-training (Section 6.3.1) and fine-tuning (Section 6.3.2). We start by describing the setup for each of the subtasks.

Masked language model (*mlm*) task We generate masked inputs following Albert n-gram masking. We mask 20% of all WordPiece tokens but do not always replace masked words with the `[MASK]` token to avoid discrepancy between pre-training and fine-tuning. We effectively replace 80% of the masked position with `[MASK]` (`[MASK/MASK]`), 10% with a random token (`[MASK/random]`), and keep the original token for the last 10% (`[MASK/original]`).

Next sentence prediction (*sop*) task We format our inputs as "`[CLS]` x_1 `[SEP]` x_2 `[SEP]`". In 50% of the case the two segments x_1 and x_2 are effectively consecutive in the text. In the other 50%, the segments are swapped.

Ponder cost We fix the time penalty factor τ empirically such that the ponder penalty represents around 10% of the total loss. To estimate the ponder cost, we discard the remainder as $R \ll N$ for sufficient values of N. Given Eq. 6.13, the ponder cost then corresponds to the total number of iterations in the sentence, which is given by $l \times T$, with T the number of tokens in the sequence and l the average iterations per token. We observe that Albert base loss converges to around 3.5. We calibrate τ such that $\tau\mathcal{P} \approx 0.35 \approx \tau \times l \times T$. We train distinct models, listed in Table 6.1, which we calibrate such that their average number of iterations per token l is respectively 3, 6, and 12. We refer to these models as *tiny*, *small* and *base* respectively.

6.3.1 Analysis of the pre-training

Analysis of the iterations We pre-train models with various configurations and observe the model mechanisms during the pre-training in Table 6.1.

Table 6.1: Average number of iterations given token types during the pre-training. For each model, we report a mean number of iterations on our development set, at the end of the pre-training.

Models	*tiny*	*small*	*base*
τ	1e-3	5e-4	2.5e-4
Max iterations	6	12	24
mlm (Acc.)	55.4	57.1	57.4
sop (Acc.)	80.9	83.9	84.3
All tokens	3.8	7.1	10.0
All unmasked tokens	3.5	6.5	9.2
`[MASK/MASK]`	5.8	10.9	16.0
`[MASK/random]`	5.8	10.9	16.0
`[MASK/original]`	4.0	7.4	10.5
`[CLS]`	6.0	12.0	22.5
`[SEP]`	2.5	7.6	8.4

We observe that the `[CLS]` token receives far more iterations than other tokens. This observation is in line with K. Clark et al. (2019) who analyze Bert attention and report systematic and broad attention to special tokens. We interpret that the `[CLS]` token is used as input for the *sop* task and aggregates

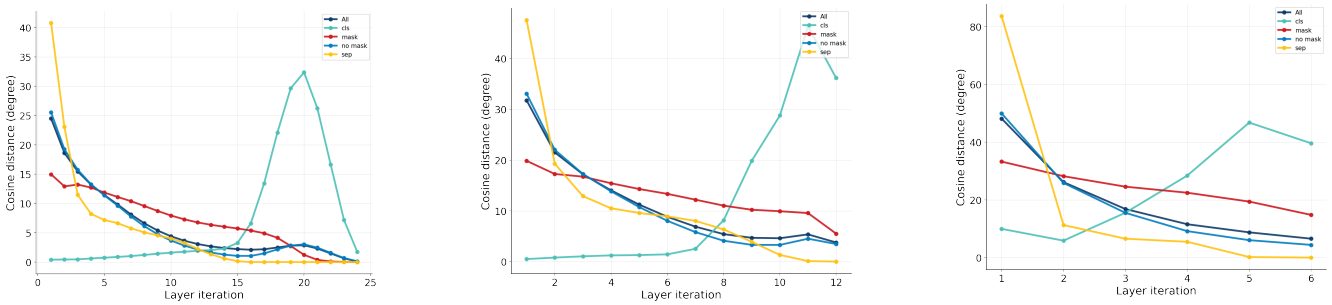
a representation for the entire input. On the contrary, `[SEP]` token benefits from usually few iterations. Again, this backs up the observation emerging from the analysis of attention that interprets `[SEP]` as a no-op operation for attention heads (K. Clark et al. 2019).

We also observe an interesting behavior from the `[MASK]` which also benefits from more iterations than average tokens. As for the `[CLS]` token, we interpret that these tokens are crucial for the *mlm* task. Looking further, we observe that the number of iterations for `[MASK/random]` and `[MASK/MASK]` is greater than `[MASK/original]`. In this case, although all tokens are targeted in the *mlm* task, `[MASK/random]` and `[MASK/MASK]` are obviously more difficult to identify.[6]

The model seems to have an intuitive mechanism and distributes iterations for tokens that are either crucial for the pre-training task or present a certain difficulty level. This also appears in line with *early-exit* mechanisms cited in Section 6.2, that adapts the number of iterations, for the whole example, to better scale to each sample level of difficulty.

6: During inference, the model cannot make the distinction between `[MASK/original]` and unmasked tokens. However, we observe in Table 6.1 that the two token types have a distinct mean number of iterations. We believe this is due to the distribution of the `[MASK]` tokens. Indeed, we follow the procedure from ALBERT and use *n-gram* masking. Therefore, `[MASK/original]` tokens tend to appear in the context of `[MASK]` tokens. This specific context increases the mean number of iterations.

Natural Fixed point We now analyze *how* the token's hidden states evolve during our model iterative transformations. At each iteration n, the self-attentive mechanism (Vaswani et al. 2017) computes the updated state $n+1$ as a weighted sum of the current states. This introduces a cyclic dependency as every token depends on each other during the iterative process. As convergence within a loopy structure is not guaranteed, we encourage the model to converge towards a fixed point (S. Bai, Kolter, and Koltun 2019).



(a) *base* model. **(b)** *small* model. **(c)** *tiny* model.

Figure 6.6: Evolution of the cosine distance between hidden states h_t^n and h_t^{n+1} from two consecutive iterations. We use our *base*, *small* and *tiny* models and measure iterations on our development set at the end of the pre-training.

We obtain this property "for free" thanks to our architecture specificity. Indeed at each iteration, the hidden state is computed as a convex combination of the previous n and current $n+1$ hidden state. The combination is controlled by λ_t^n (Eq. 6.11). If λ_t^n is closed to 0, then $h_t^n \approx h_t^{n+1}$ and by definition (Eq. 6.10, 6.12) λ_t^n will eventually be set to 0 at a certain iteration.

Figure 6.6 represents the evolution of the mean cosine similarity between two hidden states from two consecutive iterations h_t^n and h_t^{n+1}. The network indeed reaches a fixed point for every token. The `[SEP]` and tokens that are not masked converge quicker than `[MASK]` tokens. Finally, the `[CLS]` token oscillates during intermediate iterations before reaching an equilibrium.

6.3.2 Application to downstream tasks

During the pre-training phase, the model focuses on tokens either crucial for the pre-training task or that present a certain level of difficulty. Now, we study our model behavior during the fine-tuning on downstream syntactic or semantic tasks.

Control test To verify that our setup has reasonable performance, we evaluate it on the GLUE benchmark (Yau-Shian Wang, Hung-yi Lee, and Y. Chen 2019). Results from Table 6.2 are scored by the evaluation server.[7] As in Devlin et al. (2019), we discard results for the WNLI task.[8] For each task, we fine-tune the model on the train set and select the hyperparameters on the dev set using a grid search. We tune the learning rate between 5e-5, 3e-5, and 2e-5; the batch size between 16 and 32, and epochs between 2, 3, or 4. To better compare our setup, we pre-train Bert and Albert model using our configuration, infrastructure, and data.

7: https://gluebenchmark.com/leaderboard

8: See point (12) from https://gluebenchmark.com/faq.

Table 6.2: GLUE Test results, scored by the evaluation server but without the WNLI task. To facilitate the comparison, we reproduce Bert and Albert, with our pre-training dataset, infrastructure and configuration detailed in Section 6.2.3.

	Avg. Glue score
Bert-base	76.9
Albert-base	75.6
Albert-base + Adapt. Depth	75.2
Albert-small + Adapt. Depth	74.2
Albert-tiny + Adapt. Depth	72.6

We present results on the test set in Table 6.2. As expected, the average score decreases with the number of iterations. Indeed, we limit the number of computation operations performed by our model. Moreover, we build our model on top of Albert, which shares parameters across layers, thus reducing the number of parameters compared with the original Bert architecture. However, despite these additional constraints, results stay in a reasonable range. In particular, Albert-base with adaptative depth is very close to the version with a fixed depth.

Probing tasks Conneau and Kiela (2018) introduce probing tasks, which assess whether a model encodes elementary linguistic properties. Such tasks are detailed in Section 3.4.2. We consider semantic and syntactic tasks that do not introduce random replacements. In particular, a task that predicts the sequence of top constituents immediately below the sentence node (TopConst), a task that predicts the tense of the main-clause verb (Tense), and two tasks that predict the subject (resp. direct object) number in the main clause (SubjNum, resp. ObjNum). We provide examples for each of these tasks in Table 6.3.

In our setup, we fine-tune the model on the task train set and select the hyperparameters on the dev set using a grid search. We use a 5e-5 learning rate and fine-tune the epochs between 1 to 5; we use a batch size of 32. Finally, we compare in Table 6.4 the number of iterations performed for each token on the Penn Tree Bank (Marcus, Santorini, and Marcinkiewicz 1993) converted to Stanford dependencies.[9]

We provide an accuracy baseline, obtained with the same setup but using Albert without the dynamic halting mechanism. As in the previous experiment, we observe that for these tasks, our model achieves competitive performance despite computing fewer total iterations per tokens.[10]

Although all tasks achieve significant and comparable accuracies, they all require a distinct global mean of iterations. The Tense task, which can be solved from the verb only, is completed in only 5.4 iterations, while the TopConst task, which requires inferring some sentence structure, is performed in **7.2** iterations. This suggests the model can adapt itself to the complexity of the task and globally spare unnecessary iterations.

9: Since we use SentencePiece vocabulary, we assign to each piece the dependency tag from the whole token.

10: However, our model does not necessarily perform fewer computational operations. The halting mechanism requires some additional operations, and we must still perform iterations while all tokens did not stop. In practise, as detailed in Table 6.1, the maximum number of iterations may reach 6, 12 or 24 for the models *tiny*, *small* and *base* respectively, which may be more than Albert. As opposed to optimizing the model's computation efficiency, we are more interested in analyzing the patterns it learns.

Table 6.3: Examples from the tasks of the probing tasks (Conneau, Kruszewski, et al. 2018) of the SentEval benchmark that we use in our experiments. The top constituent task (**TopConst**), aims at predicting the top constituent immediately below the sentence root node. The **Tense** task aims at predicting the tense of the main clause verb. The subject and object number (respectively **SubjNum** and **ObjNum**) tasks focus on the number of respectively the subject and object of the main clause. All sentences are extracted from the Toronto Book Corpus (Zhu et al. 2015). The part-of-speech, constituency and dependency parsing information are provided with the Stanford Parser (2017-06-09 version), using the pre-trained PCFG model (Klein and Manning 2003). The top constituent task (**TopConst**) classifies sentences on the basis of their sequences of top constituents immediately below the sentence node. There are 19 classes for the most frequent top constructions, and one for all other constructions.

Sentence	**Label**
Tense	
The smell churned my stomach even faster .	PAST
I lean against the post and watch her set up .	PRES
Subj Num	
The crows circled above me .	NNS
" I imagine it 's nothing good , but speak , " the colonel said .	NN
Obj Num	
I saw the ramp leading back toward the surface .	NN
I could feel their stares like hot rays penetrating into me .	NNS
Top Const	
They obviously protect him from anything he won 't like .	S_NP_VP_.
Did it belong to the owner of the house ?	VBD_NP_VP_.

Looking at the token level, as during the pre-training (Section 6.3.1), the iterations are unevenly distributed across tokens. The model seems to iterate more on crucial tokens for the task. For SubjNum, the subj tokens achieve the maximum number of iterations, while for the ObjNum task, the obj and root token iterates more. Similarly, all tasks present many iterations on the main verb (root) that is crucial for each prediction.

	Tense	Subj Num	Obj Num	Top Const
punct (121k)	5.0	4.8	5.2	6.7
prep (101k)	4.6	4.6	5.4	6.2
pobj (98k)	4.5	4.6	5.4	5.8
det (86k)	4.5	4.6	5.1	6.1
nn (81k)	5.1	5.4	**5.8**	6.7
nsubj (80k)	**5.3**	**6.1**	**5.9**	**7.5**
amod (66k)	4.6	4.9	5.5	6.1
dobj (49k)	4.8	5.0	**5.9**	6.1
root (44k)	**5.9**	**6.1**	**6.2**	**7.9**
advmod (37k)	4.8	4.8	5.3	6.8
avg.	5.4	5.4	5.8	7.2
test Acc.	87.5	93.9	96.1	91.2
baseline Acc.	87.3	94.0	96.0	91.9

Table 6.4: Distribution of the iterations across token dependency types. We fine-tune our *base* model on each probing task. We then perform inference on the Penn Tree Bank dataset and report the number of iterations given token dependency types. In parentheses, we indicate the number of occurrences for each dependency tag. We only display the top 10 most frequent tags. We indicate in **bold** tags for which the number of iterations is above avg + std. We include a baseline accuracy which we obtain with the Albert-base version without an adaptative depth mechanism, and therefore 12 iterations were performed for each token.

6.4 Conclusion and future work

We investigated the role of the layers in deep transformers. We designed an original model that progressively transforms each token through a dynamic number of iterations. We analyzed the distribution of these iterations during pretraining and confirmed the results obtained by analyzing the distribution of attention across Bert layers, particularly the specific behavior played by special tokens. Moreover, we observed that key tokens for the prediction task benefit from more iterations. We confirmed this observation during fine-tuning, where the tokens with a large number of iterations are also suspected to be key for achieving the task.

Our experiments provide a new interpretation path for the role of layers in deep transformer models. Rather than extracting some specific features at each stage, layers could be interpreted as iterations from a convergent process. We hope that this can help to better understand the convergence mechanisms for transformers models, reduce the computational footprint or provide new regularization methods.

Analyzing the relation between compositional properties and neural structures

7

> [. . .] *how does a person answer why something happens? For example, Aunt Minnie is in the hospital. Why? Because she went out, slipped on the ice, and broke her hip. That satisfies people. It satisfies, but it wouldn't satisfy someone who came from another planet and knew nothing about why when you break your hip do you go to the hospital.* [. . .] *when you explain a why, you have to be in some framework that you allow something to be true. Otherwise, you're perpetually asking why.* [. . .]
>
> **— Richard Feynman**
> TV program *Fun to Imagine*, 1983

Compositionality is thought to be a key feature of human language. Symbolic generative theories of language indeed imply the possibility of producing an infinite number of grammatical phrases and sentences using only finite means (Chomsky 1957; Hauser, Chomsky, and Fitch 2002; Montague 1970). Humans derive phrase meaning by composing syntactic and semantic components using compositional rules (Cann 1993; Dowty 2007; Partee et al. 1984).

On the other hand, language models are trained using self-supervised objectives with no direct linguistically oriented supervision. Nonetheless, recent large pre-trained transformer models have shown striking abilities to process human language and over-perform humans on many benchmarks (Brown et al. 2020; Devlin et al. 2019). They also exhibit strong consistency on agreements (subject-verb, noun-adverb, verb-verb) which are determined by abstract structures and not just linear order of words (Gulordava et al. 2018; Linzen, Dupoux, and Goldberg 2016; Marvin and Linzen 2018; Newman et al. 2021). However, many studies point out that their compositional abilities are surprisingly limited and that they struggle to generalize to specific out-of-domain examples (Hupkes et al. 2020; Kim and Linzen 2020; Brenden M. Lake and Baroni 2018).

The ability of language models to process language without inducing exhaustive symbolic composition rules is not yet fully understood. Baroni (2019) suggest neural networks may process language using partially or different rules than humans. They emphasize human language is not fully characterized by algebraic rules. Language models might rely on less systematic phenomena such as semi-lexicalized constraints in syntax or irregular inflections. Brenden M Lake et al. (2017) explore the possibility that language models overcome their lack of compositional abilities with an exposition to huge amounts of data.

In the previous chapters, we integrated syntactic biases into neural networks. We evaluated the use of syntactic information on meta benchmarks such as SentEval (Conneau and Kiela 2018) and GLUE (Yau-Shian Wang, Hung-yi Lee, and Y. Chen 2019). However, as observed in Conneau, Kruszewski, et al. (2018), such tasks imply various linguistic phenomena. Although scores assess the model's ability to tackle the task, it is still unclear whether models rely on a shallow lexical pattern matching or an effective encoding of the syntactic structure and lexical information. Second, the use of large datasets for training (Bowman, Angeli, et al. 2015; Williams, Drozdov, and Bowman 2018) adds difficulty in disentangling the contribution of the data from the model structure.

In this chapter, we aim at better characterizing how the model structure may affect their degree of compositionality. We first review the current methods and resources to evaluate compositionality (Section 7.1). Such methods may present some limitations. In particular, they may use a text-to-text setup, which makes it difficult to disentangle the effect of the encoding and decoding parts. Other methods are also sometimes focused on a limited range of linguistic phenomena. We propose two contributions in which we compare distinct structured models, including tree-shaped models and BERT. In Section 7.2, we analyze how model structure impacts the degree of syntactic information captured by models. Using a natural language inference task, we compare the performance on sets of examples with specific linguistic properties. In Section 7.3, we intend to accurately quantify to which extent model structure is preeminent to draw compositional knowledge. We build an evaluation setup with arithmetic expressions containing specific properties. We observe how models generalize outside their domain.

7.1 Evaluating compositionality

It is undoubtedly possible to train efficient language models without prior or posterior compositional properties. However, building more compositional models is an active subject of research. Such methods seek to improve transformer models at learning compositional rules.

The first step toward building such models is to provide accurate methods to measure their compositional abilities, which is notoriously hard. First—as for every other language evaluation benchmark—creating the data is a critical step. Labeling raw data is time-consuming, and it isn't easy to precisely control the examples' property. On the other hand, generating artificial data may lead to poor lexical or structural diversity. Additionally, studies show that models might use lexical biases or shallow heuristics in the data to achieve the task (Linzen and Leonard 2018).

Many popular benchmarks use a text-to-text setup: models take raw text as input and should map it to a semantic form: SCAN (Brenden M. Lake and Baroni 2018), PCFG (Hupkes et al. 2020), CFQ (Keysers et al. 2020) and COGS (Kim and Linzen 2020). For the SCAN dataset, raw sentences should be mapped to a sequence of instructions, CFQ maps sentences to Sparql queries and COGS to semantic forms.

Another line of research proposes to examine models on a natural language inference task in which the properties of sentence pairs are specified and controlled. Bowman, Angeli, et al. (2015) analyze model compositional abilities by inferring logical relations between pairs of sentences. Such sentences are artificially generated using an artificial language based on logical statements. They compare the impact of structured models to encode these sentences with explicit latent recursive structures. In our work, we try here to better characterize the effect of encoder architectures, given the various compositional aspects. Dasgupta et al. (2018) propose a natural language inference task to evaluate the compositionality of sentences captured in embeddings. The studied properties include surface, semantic and syntactic information. McCoy, Pavlick, and Linzen (2019) propose a specifically designed dataset (**HANS**) to trick models on an NLI task. Examples are designed to exhibit biases learned by statistical models. A. Nie, Bennett, and Goodman (2019)

aim at distinguishing the use of structure versus lexical information. Models are trained given an adversarial training setup in which object, subject, or adjectives attachment are swapped in the sentence.

Other setups exist: **PAWS** is a large dataset containing sentences that have high lexical overlap without being paraphrases (Y. Zhang, Baldridge, and He 2019). The dataset is specifically designed to distinguish models which rely on lexical heuristic to solve sematic tasks. However, they do not present a breakdown given the linguistic properties of the sentence pairs and work on binary paraphrase classification. Andreas (2019) proposes a formalism to measure compositionality using similarity metrics through a communication game.

Beyond evaluating the compositional capability of models, many works aim to improve them. Some methods propose to integrate structural biases within the architecture: in particular Tree-LSTM (Tai, Socher, and Manning 2015) or in transformers with structured attention (Russin et al. 2019). Some methods also propose to adapt the pre-training or fine-tuning procedure (Furrer et al. 2020). Finally, other methods propose to complete models with modules dedicated to compositional operations (W. Liu et al. 2020; Ontañón et al. 2021).

7.2 Natural language inference

This chapter describes two experiments exploring whether models rely on explicit structure and compositional operations to capture sentence meaning. This section starts with analyzing the degree of syntactic information captured by structured models. We focus on a Natural Language Inference task (NLI), which aims to determine whether a given sentence entails a second. We embed both sentences from a pair using various structured models and link the embeddings using a similarity module that classifies the sentence pairs. We analyze how the sentences' specific syntactic and lexical properties impact the final performance.

We hypothesize that the complexity of the task requires a compositional aptitude to derive the general sentence meaning. However, recent work reveals that models can be

easily tricked by specifically designed adversarial examples. Such examples present lexical or syntactic variations that are easily understandable for humans but result in deep confusion for statistical-based models. To analyze the degree of syntactic information captured by structured models, we propose to evaluate a set of models on NLI examples presenting specific syntactic or lexical properties. Such an analysis grid allows us to distinguish the performance given models and linguistic phenomena. We use it to analyze how various structured models perform specific compositional operations.

Examples are selected given the SICK dataset (Marelli et al. 2014) which consists of an NLI task specifically designed to assess model compositional properties. Samples are built to present a rich collection of linguistic constructions. Moreover, syntactic and lexical operations are designed to require specific compositional awareness. Each sentence pair is labeled given the studied transformation. We propose categorizing transformations given the induced changes on the surface lexical form or the underlying sentence structure.

Our experiments reveal that some models are indeed better adapted to capture some linguistic aspects. Moreover, we witness that some specific sentence structures or lexicon might result in severe confusion when performing inference. In particular, the replacement of words with semantic opposites or the scrambling of words between the premise and hypothesis. For the latter, we perform additional analysis to better capture the inclination of models to generalize in such complex linguistic structures. We show that, while some cases might be addressed by carefully choosing the model structure, for others, it seems the relation of entailment can only be learned by augmenting the training set with corresponding examples.

7.2.1 Method

We map sentences s to embeddings h_s using the models exposed in Section 3.2: Bag Of Word (Bow), sequential LSTM (Seq), hierarchical models such as Tree LSTM for dependency and constituency structure (Dep and Const) and Bert (Bert-cls).

Regarding the Tree LSTM models, we obtain the dependency structures using the deep biaffine parser from Dozat and Manning (2017) and the constituency structures using the constituency neural parser from Kitaev and Klein (2018). Regarding Bert, we use the original Bert-base and the Bert-cls token as the sentence embedding.

For each model, we consider two types of embeddings vectors. We use either traditional GloVe embeddings (Pennington, Socher, and Manning 2014) or, as illustrated in Figure 7.1, contextualized vector from the Bert model (Devlin et al. 2019). When using Bert embeddings, vectors are normalized with the L2 norm for Bow model to facilitate the joint convergence of the model and Bert layers. When using GloVe embeddings, we used 300-dimensional word vectors trained on the common crawl dataset (840B tokens) with a vocabulary of 2.2M case-sensitive words.

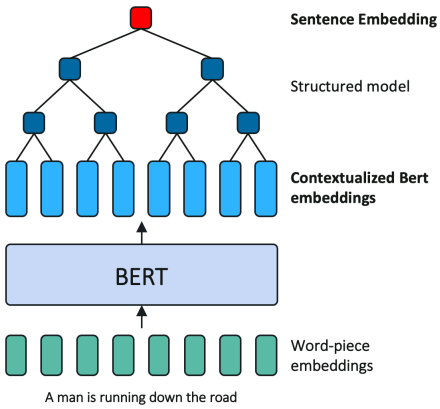


Figure 7.1: Models architecture: We use either GloVe static embedding or Bert in a *fine-tuning* configuration such that the final output layer is used as input for a structured model as detailed in the Section 7.2.1.

7.2.1.1 Similarity architecture and training objective

We choose a similar supervised framework as in Tai, Socher, and Manning (2015) and use the given train/dev/test split. We illustrate the full model in Figure 7.2. Parameters are set given results on the dev set, and results are presented on the test set. In the experiments section, results are presented given the transformation applied to each couple of sentences.

Regarding Bert, we do not use the similarity architecture, but rather directly feed the sentence pair to Bert. The prediction is based on the Bert-cls token, later processed by a linear layer and a softmax activation function.

For other encoders, a dedicated architecture predicts the similarity distribution of a pair of sentences. The similarity module takes as input a pair of sentence vectors (h_L, h_L) and computes their componentwise product $h_L \odot h_R$ and their absolute difference $|h_L - h_R|$. These features are then fed to a two-layer perceptron network (MLP) to compute the probability distribution $\hat{p}_\theta$:

$$\begin{aligned} h_\times &= h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\ h_s &= \sigma(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}), \\ \hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}), \end{aligned} \tag{7.1}$$

The KL-divergence between the predicted distribution $\hat{p}_\theta$ and the ground truth p is used as a training objective:

$$J(\theta) = \frac{1}{N} \sum_{k=1}^{N} \text{KL}(p^{(k)} \Big\| \hat{p}_\theta^{(k)}) + \frac{\lambda}{2} ||\theta||_2^2 \tag{7.2}$$

The ground truth p is a three-dimensional one-hot vector defined given the relation between the two sentences in the pair (entailment, contradiction, or neutral). During inference, the argmax of $\hat{p}$ is considered. We report the accuracy between targets and predictions to evaluate the performance.

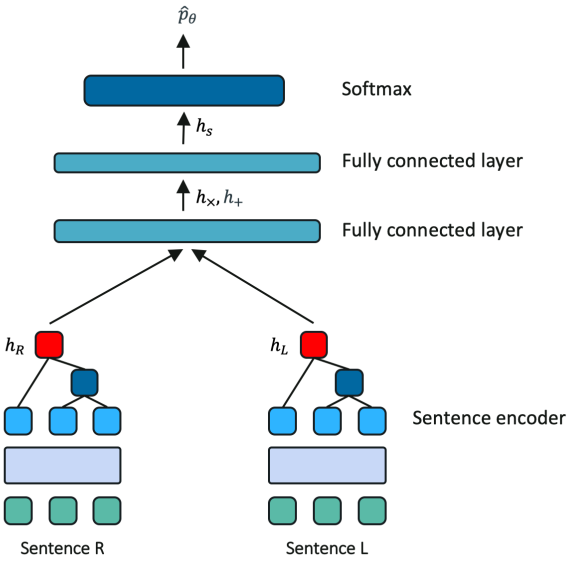


Figure 7.2: We embed sentences using various structured models. The similarity architecture uses the sentence embeddings component-wise product and absolute difference to output a probability distribution for the entailment label.

7.2.1.2 Hyper parameters setting

Parameters are fixed using the dev set. We use a batch size of 25. We use the Bert-base-cased model (Devlin et al. 2019) distributed on Huggingface (Wolf et al. 2020) with an embedding size of 768. For Bert layers, we use 0.10 weight decay and fix the learning rate to 2e-5. For other layers, the learning rate is fixed to 0.025 with a weight decay of 1e-4. Layers are initialized using a glorot distribution (Glorot and Bengio 2010). Biases are initially set to 0 for tree structure models and 1 for Seq model. When applicable, the model hidden-size is fixed to respectively 150 and 300 when using GloVe and Bert emeddings. The hidden-size of the similarity module is set to 50. Models are trained during a maximum of 20 epochs using the Adagrad optimizer (Duchi, Hazan, and Singer 2011) and AdamW for Bert (Loshchilov and Hutter 2019). When no improvement is observed on the dev set for 3 consecutive epochs, the training is stopped. Results are reported on the test set.

7.2.2 Linguistic phenomena experiment

7.2.2.1 SICK Data

The SICK dataset (Marelli et al. 2014) consists of 9,840 sentence pairs which have been manually annotated to assess whether the first entails the second. The original sentences are sampled from the 8K Image Flickr dataset (Hodosh, Young, and Hockenmaier 2013) and the SemEval 2012 STS MSR Video Description dataset (Agirre et al. 2012). These two datasets contain sentences describing the same picture or video. The sentence pairs are then transformed through a 3-step process: normalization to remove unwanted linguistic phenomena; expansion to obtain sentences with specific characteristics; pairing expanded sentences with normalized sentences (and pairing both normalized sentences from the pair). The dataset is already split into train/dev/test containing respectively 4,506/505/4,979 samples. Sentence pairs have been manually transformed to include specific syntactic or lexical properties detailed in Table 7.1. The dataset is freely available for research purposes.[1] Moreover, some additional indexes detail the transformation that was applied to one of the sentences from each pair. We use this information to

1: https://zenodo.org/record/2787612#.Xjh2qxfjJ24

Table 7.1: Sick expansion rules: Detailed transformations applied to generate the SICK dataset. Transformations are categorized given their impact on the sentence surface form. The null transformation refers to sentence pairs for which no expansion transformation was performed: we only pair two normalized sentences describing the same picture in the original dataset. For every other transformation, the final sentence pair is generated out of a single original sentence transformed with normalization and expansion operations.

Transformations		Examples	
	Lexical transformations		
null	No transformation	Three boys are jumping in the leaves	Children in red shirts are playing in the leaves
so	Replace words with semantic opposites	A **man** is rowing a boat	A **woman** is rowing a boat
lex	Replace words with synonyms	A young **boy** is jumping into water	A young **kid** is jumping into water
det	Replace quantifiers	**The** surfer is riding a big wave	**A** surfer is riding a big wave
	Syntactic clause expansion		
aa	Add modifiers	A deer is jumping a fence	A **wild** deer is jumping a fence
expn	Expand agentive nouns	**Some people playing rugby** are tackling each other	**Rugby players** are tackling each other
expa	Turn adjectives into relative clauses	A **cute** panda is lying down	A panda **that is cute** is lying down
expc	Turn compounds into relative clauses	The woman is frying a **chop of breaded pork**	The woman is frying a **breaded pork chop**
	Global transformations		
od	Change determiners with opposites	**A** dog is barking	**There is no** dog barking
inv	Insert a negation	The boy is playing the piano	The boy is **not** playing the piano
top	Turn active sentences into passive	A man is driving a car	The car is being driven by a man
ws	Scramble words	The **turtle** is following the **fish**	The **fish** is following the **turtle**

perform a detailed breakdown of the performance of various models given the transformation. We divide the transformations into three categories, given the degree of induced alteration on the sentence surface form.

- **Lexical transformations** The first set of transformations induces local changes. Only individual words are modified. Given the modification, the sentence meaning can be preserved or altered. We expect every model to be robust to such changes.
- **Syntactic clause expansion** In this set of alterations, we add, remove or modify sub-trees of the sentence structure. Such transformation has a low impact on the syntactic tree structure but can deeply transform the surface form of the sentence. We expect strongly

structured models to be more robust to such transformations.

- **Global transformations** Finally, we consider a set of modifications that impact both the surface and syntactic form.

We also considered the possibility of exploiting the HANS dataset (McCoy, Pavlick, and Linzen 2019), which comprises 30,000 sentence pairs automatically generated with specific linguistic properties. Similarly to the SICK dataset, HANS aims at identifying models relying on shallow heuristics to perform natural language inference and includes a larger set of transformations. However, it does not aim to characterize model compositional abilities, as it generates sentences based on a set of templates that share a similar compositional scheme. From our point of view, it is less adapted for a fine-grained analysis of compositional knowledge.

7.2.2.2 Lexical transformations

In the following sections, we compare how structured models behave when facing such sets of transformations. Results for lexical transformations are reported in Table 7.2. We observe a distinction between **so** and **lex** transformations which replace individual words with respectively antonyms or synonyms. For **so** transformations, models show difficulty in propagating lexical transformation into the final sentence embedding and adjusting the meaning of the sentence. For **so** transformations, we also observe that the use of contextualized embeddings seems to even lower the results of structured models compared to the use of static GloVe embeddings.

Models may be sensitive to lexical variations Some pairs seem difficult to predict for Bert-cls and Bow models. For example, the following sentence pair is almost systematically wrongly classified on all five runs: "A little **boy** and a woman wearing a yellow shirt are getting splashed by a city fountain" → "A little **girl** and a woman wearing a yellow shirt are getting splashed by a city fountain". For Bow, we can extrapolate that the fact that the subject includes two people induces some confusion in the output representation. It is less clear for Bert-cls.

Table 7.2: Accuracy on the test set for SICK-E task for pair samples with lexical transformations. We report a mean over 5 runs (standard deviations in parentheses). The best results for a given embedding are in **bold**. The best results overall are underlined. The null transformation consider pair obtained out of two different original sentences. The two sentences can differ given multiple, uncontrolled lexical or syntactic phenomena. Therefore, the performance is usually lower for this category.

	No transformation (null)	**Replace words with semantic opposites (so)**	**Replace words with synonyms (lex)**	**Replace quantifiers (det)**
N	2,558	461	399	118
		Glove embeddings		
Bow	81.6 (0.9)	49.3 (3.6)	57.3 (2.7)	82.0 (5.1)
Const	84.6 (0.2)	66.5 (2.5)	76.5 (1.6)	98.6 (0.9)
Seq	83.1 (1.2)	63.7 (6.8)	74.1 (3.9)	**98.8** (0.9)
Dep	**84.8** (0.5)	**69.6** (1.2)	**76.5** (2.4)	98.6 (0.7)
		Bert embeddings		
Bow	77.5 (1.2)	26.9 (6.0)	80.2(4.3)	93.6 (5.9)
Const	76.0 (2.7)	53.6 (3.5)	77.8 (3.0)	96.8 (2.4)
Seq	81.5 (1.3)	46.9 (3.3)	**83.6** (2.1)	**100.0** (0.0)
Dep	82.7 (0.9)	57.6 (2.0)	80.8 (1.2)	99.3 (0.6)
Bert-cls	**86.1** (1.4)	**68.8** (7.4)	77.4 (3.6)	98.3 (0.9)

The Dep model seems particularly sensitive to transformations operated on the sentence's main verb. For example, the following sentence pair is better classified by Dep and Seq models: "A woman is **applying** cosmetics to her eyelid" → "A woman is **removing** cosmetics from her eyelid".

7.2.2.3 Syntactic clause expansion

This set regroups transformations involving syntactic expansion. All models show a strong robustness to adding modifiers (**aa** expansion). When only using GloVe embeddings, expanding agentive nouns (**expa**), turning adjectives into relative clauses (**expn**) and turning compounds into relative clauses (**expc**) expansions seem to be better captured by the tree-structured model. However, using Bert embeddings reduces the gap between models, which suggests some syntactic information is captured in contextualized embeddings for such transformations. It is particularly explicit for the Bow model as the performance using GloVe or Bert word embedding deeply differs for all transformations.

Some idiomatic structures are only captured by pre-trained models Some pairs benefit from the use of Bert while

Table 7.3: Accuracy on the test set for SICK-E task for pair samples with syntactic clause expansion. We report mean over 5 runs (standard deviations in parentheses). Best results for a given embedding are in **bold**. Best results overall are underlined.

	Add modifiers (aa)	**Expand agentive nouns (expn)**	**Turn adjectives into relative clauses (expa)**	**Turn compounds into relative clauses (expc)**
N	138	12	58	19
		Glove embeddings		
Bow	77.4 (4.7)	53.3 (8.5)	60.7 (8.5)	64.2 (10.7)
Const	90.4 (0.8)	**71.7** (11.3)	87.2 (0.8)	86.3 (7.1)
Seq	90.0 (2.5)	66.7 (9.1)	85.2 (3.6)	86.3 (7.1)
Dep	**93.8** (1.5)	48.3 (13.3)	**93.8** (1.8)	**89.5** (3.3)
		Bert embeddings		
Bow	93.6 (4.3)	63.3 (4.1)	95.2 (6.5)	**100.0** (0.0)
Const	88.0 (4.5)	51.7 (14.3)	84.1 (2.5)	88.4 (6.1)
Seq	96.8 (1.3)	78.3 (6.7)	**97.6** (1.8)	98.9 (2.1)
Dep	94.8 (0.5)	**81.7** (6.2)	95.5 (2.3)	92.6 (2.6)
Bert-cls	**97.5** (0.7)	76.7 (8.2)	94.1 (4.6)	93.7 (6.1)

structure does not help with the prediction. For example: "Someone is feeding an animal" → "Someone is giving food to an animal" is always mapped to the wrong label with Dep and Const models, while Bert-cls completes a perfect prediction. The Bert pre-training process can correctly match the semantic similarity between "feeding' and "giving food to", while compositional operations following a given structure do not.

7.2.2.4 Global transformations

Bert does not systematically encode structure An example from the word scrambling **(ws)** transformation illustrates the advantage of structure in shallow cases: "A surfer is leaning the surfboard against a wall" → "A surfer is leaning on a surfboard". Bert-cls and Bow consistently predict the wrong label, while all other models correctly classify the pair in almost all runs.

Table 7.4: Accuracy on the test set for SICK-E task for pair samples with syntactic clause expansion. We report a mean over 5 runs (standard deviations in parentheses). The best results for a given embedding are in **bold**. The best results overall are underlined.

	Change determiners with opposites (od)	**Insert a negation (inv)**	**Turn active sentences into passive (top)**	**Scramble words (ws)**
N	290	172	125	157
		Glove embeddings		
Bow	89.0 (2.4)	83.4 (7.9)	**89.4** (3.3)	55.5 (3.9)
Const	96.7 (1.5)	96.7 (1.0)	85.9 (2.7)	60.0 (3.7)
Seq	**97.4** (0.3)	**97.1** (1.5)	84.0 (4.9)	**62.9** (3.7)
Dep	97.3 (1.0)	96.9 (0.5)	87.2 (4.3)	62.2 (3.1)
		Bert embeddings		
Bow	94.8 (1.6)	94.8 (1.7)	88.5 (8.2)	54.0 (5.8)
Const	94.5 (1.6)	94.3 (2.2)	82.7 (15.0)	61.8 (5.9)
Seq	**97.2** (0.3)	**97.4** (0.9)	93.3 (5.0)	55.8 (4.2)
Dep	96.0 (0.8)	96.7 (0.6)	94.2 (1.9)	**64.6** (3.7)
Bert-cls	96.3 (1.7)	94.4 (5.5)	**94.2** (2.0)	47.0 (11.4)

7.2.3 Focus on the word scrambling transformation

As observed in Section 7.2.2, all models poorly perform when facing complex transformations such as word scrambling. Such category contains distinct transformations, including switching the arguments of a transitive verb, mixing modifiers, exploiting verb transitive/intransitive alternations and exploiting homonymy and polysemy. From our hypothesis, such transformation requires heavy compositional ability. Therefore, we propose to investigate deeper the possibility of addressing such model limitations from an architectural point of view. However, to keep trackable results and analysis, we restrict the possible spectrum of transformations and limit ourselves to switching the arguments and mixing modifiers operations. Moreover, we only consider GloVe word vectors models other than Bert.

7.2.3.1 SWAP dataset

The dataset introduced in Y. Nie, Yicheng Wang, and Bansal (2019) is an NLI dataset composed of automatically generated sentence pairs whose logical relations cannot be extracted from lexical information alone. We refer to the latter as the SWAP dataset. The dataset is decomposed following two types of adversarial data for which the semantics of the

sentence is changed by keeping the same lexicon and only modifying its compositional structure. The two transformations are close to the one conducted for the word scrambling **(ws)** on the SICK task, in particular the switching of the arguments and mixing modifiers operations.

SOSWAP The transformation inverses the subject and object in the sentence. For example, the following sentence is modified as follows: "A child is pulling a woman on a sled in the snow." → "A woman is pulling a child on a sled in the snow.". The obtained sentence pair is labeled as *contradictory*. The dataset contains 971 examples generated given this transformation.

ADDMOD The transformation changes the adjective affection from the subject to the object. For example: "A cat sits alone in dry yellow grass." → "A yellow cat sits alone in dry grass.". The obtained sentence pair is labeled as *neutral*. The dataset contains 1,783 examples generated given this transformation.

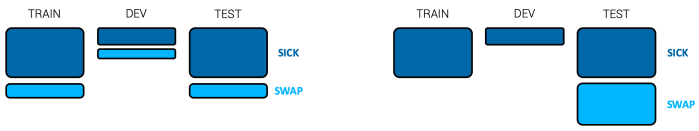


Figure 7.3: Concatenation of the dataset: *(right)* we distribute the examples from the SWAP dataset following a stratified train/dev/test split among the corresponding splits of the SICK dataset. *(left)* We only include the SWAP dataset in the SICK test set.

As the dataset size is limited, we concatenate the SICK and SWAP datasets. We experiment with two distinct concatenation strategies illustrated in Figure 7.3. We use a stratified train/dev/test split and distribute the examples among the corresponding split from the SICK dataset. In this configuration, the model is presented with specific examples of the transformation during training. In the second configuration, all examples are included in the SICK test set. During training, the model is not presented with any of the considered examples. Here, we aim to evaluate the ability of the various models to generalize to unseen typologies of examples. Regarding the training setup, we keep the method detailed in Section 7.2.1.1 as well as the hyper-parameter setting.

7.2.3.2 Results analysis

Table 7.5: Accuracy on the test set for SICK-E and SWAP task given the two proposed aggregation strategies. We report a mean over 5 runs (standard deviations in parentheses). The best results for a given embedding are in **bold**. The best results overall are underlined.

	SWAP			SWAP (only SICK training examples)		
	ALL	SOSWAP	ADAMOD	ALL	SOSWAP	ADAMOD
Test Size	6,356	485	892	7,733	971	1,783
Bow	72.5 (1.1)	40.7 (4.1)	82.6 (3.2)	53.2 (4.1)	**16.0** (10.4)	**16.5** (10.7)
Const	**85.5** (0.3)	**90.5** (2.3)	**97.2** (1.2)	54.3 (1.1)	0.2 (0.2)	6.0 (3.4)
Seq	83.7 (0.9)	89.9 (3.3)	95.0 (1.2)	53.5 (0.3)	0.2 (0.3)	0.8 (0.2)
Dep	84.3 (0.7)	78.4 (5.1)	94.1 (0.7)	**56.5** (0.5)	0.3 (0.4)	12.0 (2.6)
Bert	**86.5** (1.6)	**91.4** (5.0)	**96.6** (2.1)	**53.9** (0.6)	**1.0** (0.5)	**1.0** (0.7)

We present the accuracy scores on the test set in Table 7.5. In the two concatenation strategies, we present the accuracy of the entire test set and filter on the SOSWAP and ADAMOD test pairs.

The transformations in the SWAP dataset are much more limited than in the SICK dataset which includes additional transformations such as verb transitive/intransitive alternations or exploiting homonymy and polysemy. Nevertheless, when no example is included in the training set, the accuracy scores seem in line with the one for the word scrambling **(ws)** transformation for the SICK dataset in Table 7.4.

For both transformations, all models fail to predict the correct label when not exposed to the transformation during the training. Only the Bow model stands out but with a substantial standard variation. However, the Dep model seems not entirely tricked by the ADAMOD transformation. Even more remarkable, Bert is completely fooled as well. From these specific swapping examples, we assume that despite explicit structure assimilation in the model, they may not be able to generalize to a specific topology of unseen examples. This may also contribute to the lower results for the word scrambling **(ws)** on the SICK task.

7.3 Arithmetic expressions evaluation

So far, we used data generated via a semi-automated procedure, which enables precise control of the property of the examples. On the other hand, generating artificial data

may lead to poor lexical or structural diversity. Studies show that models might use lexical biases or shallow heuristics in the data to achieve the task (Linzen and Leonard 2018). Consequently, shallow lexical and structural pattern matching operations may be difficult to distinguish from effective compositional operations.

Here, we aim to examine the compositional properties of neural architectures without being affected by lexical phenomena. Therefore, we propose a setup for which neural models compose representations for sentences that do not include words: arithmetic expressions. Arithmetic defines a self-contained universe which can be described using a limited set of symbols and composition rules. This makes it easy to build specific examples with isolated properties. By extension, we may evaluate neural models' compositional abilities. Indeed, numerically evaluating formal expressions theoretically requires capturing the formal rules of arithmetic. We use the work from Hupkes et al. (2020) and consider aspects of compositionality that may also be applicable for language: localism, substitutivity, productivity, and systematicity.

Neural networks may be properly trained to solve mathematical expressions. A line of work outlines that pre-trained language models or static word embeddings capture scales and notions of numeracy (Naik et al. 2019; Sundararaman et al. 2020; Thawani et al. 2021; Wallace et al. 2019; Z. Zhang et al. 2020). Beyond representing numbers, further work also analyzes the ability of models to perform basic mathematical reasoning (Dua et al. 2019; Geva, Gupta, and Berant 2020; Saxton et al. 2019) or solve mathematical expressions (Lample and Charton 2020).

Many of these related works mix raw text and numeracy and focus on the ability of language models to handle numeric representations. However, our problem is different. Therefore, we introduce **CobA**, a **Co**mpositionality **b**enchmark using **A**rithmetic. The dataset consists of simple arithmetic expressions combining natural integers with addition and multiplication operators. For example, $(5+4)\times 2$. We generate partitions of the dataset with specific in-domain and generalization sets, designed to evaluate the model's ability for each compositional aspect: localism, substitutivity, productivity, and systematicity.

In Section 7.3.1, we detail the data generation process and

the distinct dataset partitions. In Section 7.3.1, we present our training and evaluation setup. We also present our main results on the CobA generalization set. In Section 7.3.3, we perform an in-depth study to better analyze how the choice of hyper-parameters might benefit specific abilities and how the complexity of expressions impacts model performance.

7.3.1 Dataset description

7.3.1.1 Generation procedure

Using an automatic procedure, we generate arithmetic expressions. First, we generate a natural integer between 1 and 100, for example, 34. We then decompose it into the addition or multiplication of two other integers, such as 2×17. We then recursively decompose each integer in the new expression as the product or sum of two integers or keep it unchanged, with a probability p.[2]

2: We implement specific rules for numbers that are prime and cannot be decomposed with multiplication. We set the probability p to expand, by default, at 0.5.

As in Lample and Charton (2020), we use prefix notation (also known as normal Polish notation). The arithmetic expression $2 \times (14 + 3)$ is represented as the sequence $\times 2 + 14\ 3$. This notation avoids the use of parenthesis and therefore leads to shorter expressions. We assign each symbol—natural integer or operator sign—to a given token. We distinguish between two distinct left and right-hand sides of an operator for each expression. For example, we make the distinction between the expression $14 + 3$ and $3 + 14$. We generate over 2.5M unique expressions with between 0 and 9 operators given this procedure.

7.3.1.2 Partitioning the dataset

Our dataset aims at evaluating model compositional properties. We take inspiration from the procedure proposed in SCAN (Brenden M. Lake and Baroni 2018; Loula, Baroni, and Brenden M. Lake 2018). We carefully select expressions to create partitions (in-domain and generalization sets) from the dataset. We split them such that in-domain and generalization sets have different distributions. It is impossible to infer generalization examples without fully capturing the properties ruling this specific aspect in the in-domain set. We thus compare the ability of models to perform *out-of-domain*

Table 7.6: Key statistics given each partition of the dataset. We report the figures for the in-domain / generalization set for each statistic. We express the "Expressions with odds and evens", "Swapped-expressions in-domain" and "sub-expressions in-domain" as the proportion of expressions verifying the property for each set. The statistics that are determinants for the aspect studied in a given partition appear in **bold**.

Partitions	Random	Systematicity	Productivity	Localism	Substitutivity
Mean value	66.3 / 66.2	67.0 / 68.2	66.4 / 65.3	66.1 / 67.4	66.1 / 66.4
Min number of operators	0 / 0	2 / 2	**0 / 4**	2 / 2	2 / 2
Mean number of operators	2.8 / 2.8	2.6 / 2.5	**2.8 / 6.5**	2.8 / 2.7	2.8 / 2.7
Max number of operators	3 / 3	3 / 3	**3 / 9**	3 / 3	3 / 3
Expressions with odds and evens (%)	85.6 / 85.4	**0.0 / 100.0**	85.6 / 98.3	85.7 / 85.3	85.7 / 85.1
Swapped-expressions in-domain (%)	1.3 / 1.6	10.0 / 0.0	1.2 / 0.0	1.2 / 1.5	**0.0 / 0.0**
Sub-expressions in-domain (%)	2.2 / 2.3	4.2 / 1.3	2.1 / 0.7	**0.0 / 00**	1.0 / 1.1

generalization. We distinguish between the model learning shallow heuristics such as local pattern matching and the one learning true compositional operations.

We build partitions given the work from Hupkes et al. (2020), which distinguishes sub-properties within compositionality: **Localism**, **Substitutivity**, **Productivity** and **Systematicity**.[3] Each of the partitions detailed below has a key statistic distribution and is designed to evaluate a model's performance along a given aspect. Each partition contains an in-domain set of 24,000 expressions and a generalization set of 12,000 expressions. We present other key statistics for the partitions in Table 7.6.

3: Hupkes et al. (2020) also enumerate the over-generalisation aspect which evaluate the accommodation to exceptions. However, we find it complex to adapt this property for our specific dataset and therefore discard it in this work.

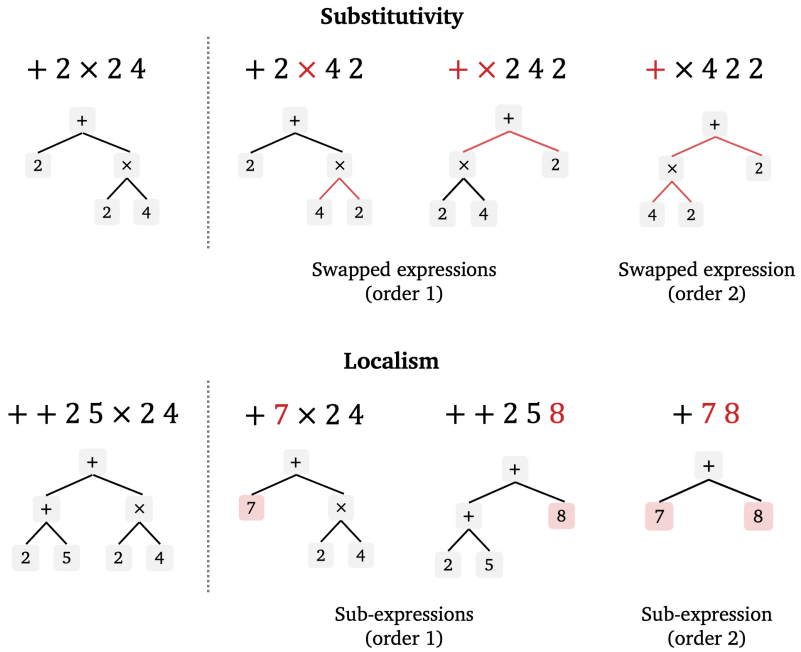


Figure 7.4: Generation of tuples of expressions for probing substitutivity and localism. For localism, we can deduce expressions from the expression seed by evaluating sub-components. For substitutivity, we can deduce expressions from each other by swapping operators' left and right-hand sides. For the clarity of the illustration, we use the infix form for the expressions.

Random is a regular training procedure. We split the dataset randomly without any specific control during the selection of the expressions. While in-domain and generalization examples are all distinct, they share the same distributions and have similar underlying characteristics.

Systematicity evaluates the recombination of known parts to form new sequences. We build the partition using the distinction between odd and even natural integers. The training set contains expressions with either only odd or even numbers, for example $2 \times 4 + 8$. or $3 + 5 + 7$. The test set contains expressions with both even and odd numbers such as $3 + 2 \times 5 + 4$.

Productivity evaluates the extrapolation to longer sequences. We train the model on expressions with up to 3 operators. We then evaluate the model on longer expressions with up to 9 operators.

Substitutivity evaluates the robustness towards the introduction of synonyms. In our work, we interpret this definition as the robustness towards paraphrases and evaluate the ability of models to perceive an operator's commutative property. We organize our dataset as a collection of "swapped expressions". Swapped expressions are tuples of expressions with the same value and only differ by swapping each operator's left and right-hand sides. We illustrate this swapping organization in Figure 7.4. During training, we only expose the model to a single expression per tuple. Therefore the model cannot learn the commutative property from shallow pattern matching. During the evaluation, we evaluate the model's commutative ability by comparing couples of predicted values for expressions from the same tuple.

Localism evaluates the recursive evaluation of smaller constituents before larger constituents. We also organize our dataset as a collection of "sub-expressions". Sub expressions are tuples of expressions with the same value that only differ by the level of decomposition of the expressions as illustrated in Figure 7.4. We use the same training and evaluation protocol as for Substitutivity.

7.3.2 Experiments

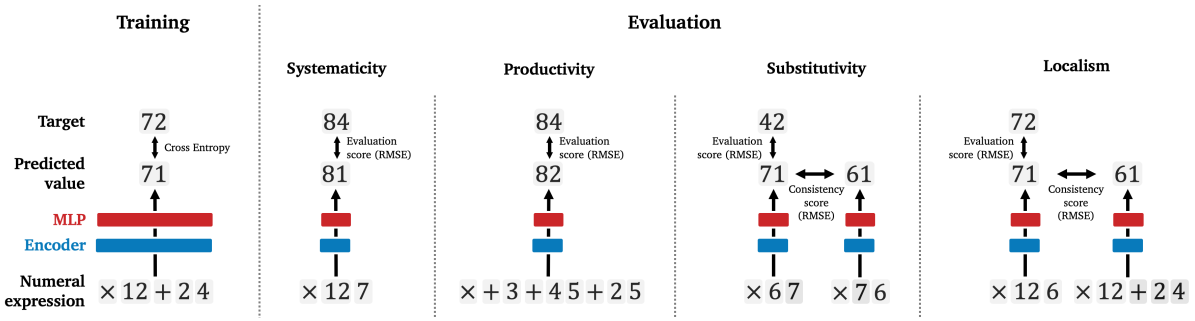


Figure 7.5: Illustration of the train and evaluation setup for assessing model compositional properties using the CobA dataset. We train the model to evaluate in-domain expressions. We then infer expressions from the generalization set. The dataset includes partitions for **Localism**, **Substitutivity**, **Productivity**, and **Systematicity**.

We train the model using a classification objective. Given an arithmetic expression, the model predicts its value among the 100 possible integers. This setup makes it possible not to use a complex decoder module; a simple probe is sufficient. The architecture is decomposed as follows: first, an encoder maps the arithmetic expression to an embedding vector. Then, a two-layer perceptron followed by a softmax outputs a probability distribution. We train the model by minimizing a cross-entropy loss.

7.3.2.1 Encoder architectures

As in Section 7.2, we use BoW, sequential LSTM, N-ary tree LSTM. We also consider transformer architectures. We derive two simple encoders from the architectures of Bert (Devlin et al. 2019) and Albert (Lan et al. 2020). We use the [CLS] token's hidden state from the last layer as the expression embedding. We initialize our models randomly and train them from scratch. Their architectures are light compared with standard transformer scales: we use a hidden size of 128, 6 hidden layers, and 8 attention heads. This represents 1.2M parameters for Bert and 300k for Albert since parameters are tied across layers. As observed in Csordás, Irie, and Schmidhuber (2021) and Ontañón et al. (2021), transformers' positional encoding are particularly important for this task. We use the method from Wallace et al. (2019) and add some random padding at the beginning of the input so that the encoder does not solve the task by overfitting the absolute position of the symbols.

7.3.2.2 Training configuration

We design all encoders comparable, with roughly the same number of parameters (1.2M), as detailed in Table 7.7. We also use the same hidden and embedding size range for all encoders: 256 for LSTM-based encoders and 128 for transformer-based encoders. We use the same optimization procedure for each model. We train all models using the AdamW optimizer (Loshchilov and Hutter 2019) with a $1e^{-3}$ learning rate, 1 epoch warm-up with polynomial decay and a batch size of 100. For each partition, we separate the in-domain set between a train and dev set using a random 90/10% split. We measure the RMSE between the expression predicted and the true value on the dev set. We stop the training when no improvement is made for 5 consecutive epochs or after a maximum of 100 epochs.[4] We train all models on an Nvidia 2080 Ti GPU. The training time is around 10 minutes per partition and model. We set parameters given the literature on the subject and do not perform a hyper-parameter search.

4: For some runs, we observed that the initialization of the model weights prevented any decrease of the loss. In such cases, we immediately stopped the run and relaunched it with a new random seed.

Regarding the natural integer embeddings, we use the DICE method (Sundararaman et al. 2020). The method uses a deterministic approach to construct natural integer embeddings. It obtains state-of-the-art results on evaluation benchmarks (Wallace et al. 2019). We do not update natural integer embeddings during training. For operators and specific tokens such as CLS or SEP tokens, we initialize embeddings randomly (with the same scale) and update them during training.

7.3.2.3 Evaluation setup

We evaluate our models and report metrics from the generalization set. We illustrate the evaluation setup in Figure 7.5. For the random, systematicity, and productivity partitions, we compute an evaluation score as the mean RMSE between each expression's true and predicted value.[5] For the localism and substitutivity partitions, we refine the evaluation procedure to take advantage of the additional paired structure of the partition described in Section **??**. We compute both an *agreement* score as the mean RMSE between the predicted values from the two expressions of each pair and an *evaluation* score as the RMSE between the predicted and true value of the expression. We report the mean between the *agreement* and *evaluation* scores. This score reflects the consistency of

5: The root-mean-square error (RMSE) between a predicted vector $\hat{y}$ and a reference y of dimensions n is defined as $RMSE(\hat{y}, y) = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$.

Table 7.7: Compositionality evaluation. We report metrics from the generalization set. For the random, systematicity and productivity partitions, we report the evaluation score, which is the RMSE between the true and the predicted values. For the localism and substitutivity partitions, we report the mean between the evaluation and consistency score. For each metric, we report the mean value over 4 runs (standard deviation in parentheses).

Encoders	# params ($\times 1e^3$)	Random	Localism	Systematicity	Productivity	Substitutivity
Random	—	39.5 (0.1)	40.1 (0.1)	40.0 (0.2)	39.1 (0.2)	40.0 (0.2)
BoW	68	28.7 (8.1)	20.6 (4.8)	37.0 (0.2)	38.3 (5.4)	19.0 (0.5)
LSTM (uni)	595	3.5 (0.4)	7.3 (0.8)	2.7 (0.3)	14.1 (0.4)	2.4 (0.3)
LSTM (bi)	1,186	2.2 (0.2)	8.6 (1.0)	2.5 (0.3)	13.5 (1.1)	1.3 (0.1)
LSTM (tree)	1,012	5.4 (0.1)	8.9 (0.1)	7.4 (0.6)	15.0 (0.5)	4.9 (0.3)
Transformer (BERT)	1,290	3.6 (1.1)	11.9 (0.7)	20.6 (4.9)	30.2 (1.8)	2.3 (0.8)
Transformer (ALBERT)	315	6.9 (1.5)	13.0 (0.9)	16.6 (6.6)	25.9 (3.1)	4.8 (1.0)

the model's predictions between two expressions as well as its ability to predict the true value. It, for example, discards trivial models which always predict the same value or model accurately evaluating one expression of the pair but failing for the other.

7.3.2.4 Results

Table 7.7 presents the results on the generalization set. We use two baselines: one that randomly predicts the value of any expression and the BoW model. We use the RMSE to compare the models. The lower it is, the better are predictions on average.

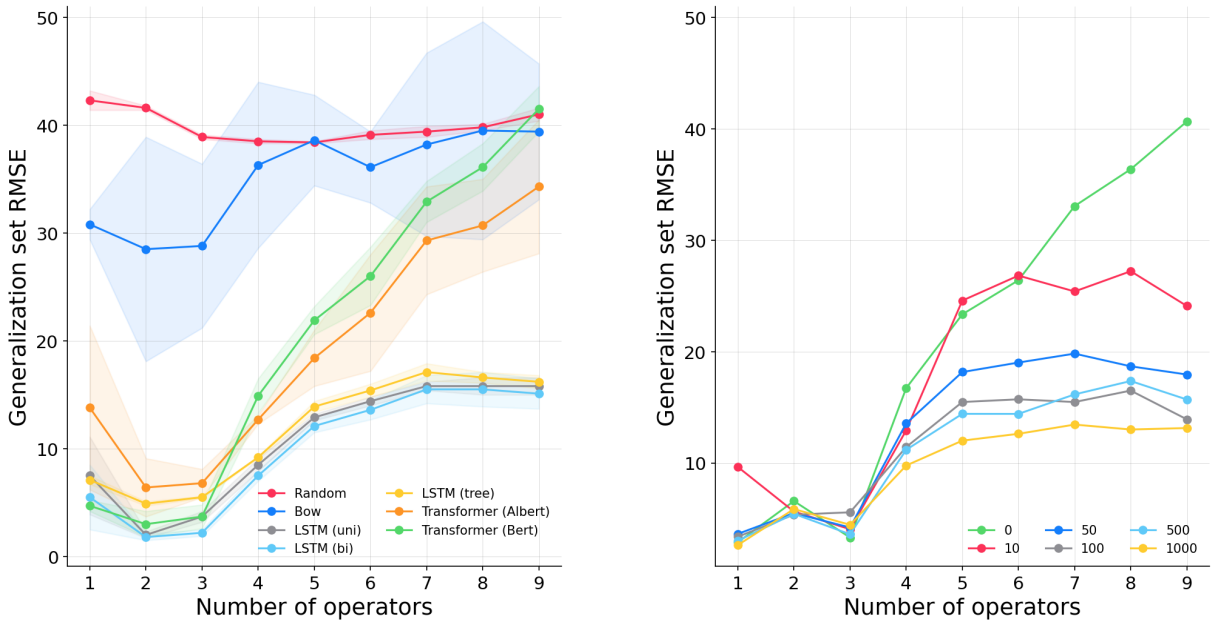
By a small margin, BoW outperforms the random guessing baseline. This suggests that the task requires accounting for the expression structure. Lexical information may provide insights for solving the task: an expression containing numbers such as a 56 and 43 is more likely of being equal to a high value such as 89 than an expression containing only a 2 and a 3. However, local information alone may not be sufficient to solve the task since expressions with a high overlap may greatly differ in value. For example, the expressions $3 + 3$ and 3×3 contain the same symbols but are not equal since they used different mathematical operators. Models can generalize to examples with similar distributions. On the random partition, all the models indeed achieve low RMSE, significantly lower than the baselines. Models are also robust toward the introduction of paraphrases since scores

on substitutivity and random partitions are similar. For other partitions, results are more contrasted.

In general, encoders relying on LSTM cells outperform transformers. For productivity and systematicity, sequential models strongly outperform models using fixed-length context. Surprisingly, sequential LSTMs constantly outperform tree LSTMs, despite having fewer structural biases.

Regarding transformers, the RMSE highly deteriorates for productivity. We confirm their known limitation in terms of productivity. We also observe transformers stumbling upon systematicity. Finally, we observe the benefit of tying parameters. ALBERT uses the same architecture as BERT, except that weights are tied across layers. ALBERT achieves results comparable or above BERT despite using far fewer parameters.

7.3.3 In-depth analysis



(a) **(b)**

Figure 7.6: Evolution of the RMSE on the productivity generalization set given the number of operators. (a) We report the mean evolution over 4 runs for each encoder (standard deviation in light). As detailed in Table 7.6, expressions from the in-domain set have 2.8 operators on average. (b) For the BERT-based encoder, we expose the model to a proportion of generalization examples during training.

Since we use generated arithmetic expressions, it is easy to control their fine-grained properties. We investigate further the influence of parameters for each partition. As part of

our study, we observe how the complexity of the examples impacts compositional abilities and how we can enhance them by modifying model hidden size or exposing models to generalization examples during training.

7.3.3.1 Impact of the expression's complexity

Complexity of compositional operations As observed in Table 7.7, all models perform reasonably well on the random partition. However, this performance might be heterogeneous across examples. We decompose the examples according to the type of operations involved. We consider expressions containing at least one addition sign (**Add**), at least one multiplication sign (**Mul**), only addition sign(s) (**Only Add**), only multiplication sign(s) (**Only Add**) and at least one multiplication and addition sign (**Add and Mul**). We present the performance given this stratification in Figure 7.7a.

Arithmetic expressions involving at least one addition operator obtain better results. Multiplications, on the other hand, tend to make the task harder. In expressions that involve addition and multiplication, there may be cases where multiplication should take precedence over addition, and for which computation order matters. Surprisingly, these expressions reach performance in line with expressions containing only one operator type.

Number of operators for productivity Productivity is notoriously hard (Baroni 2019; Hupkes et al. 2020; Kim and Linzen 2020). We also observe that neural networks struggle to generalize to longer expressions in our main results in Table 7.7. In Figure 7.6a, we decompose the productivity generalization set according to the number of operators per expression and plot the evolution of the RMSE. In line with intuition, performance declines as the number of operators grows. This evolution is not uniform across architectures: LSTM architectures generalize better to long sequences.

Number of swaps For substitutivity, we organize the dataset given tuples of swapped expressions. During evaluation, we pair each expression with an expression from the same tuple and we compare the predicted value between the two. As illustrated in Figure 7.4, we can rank all expression pairs

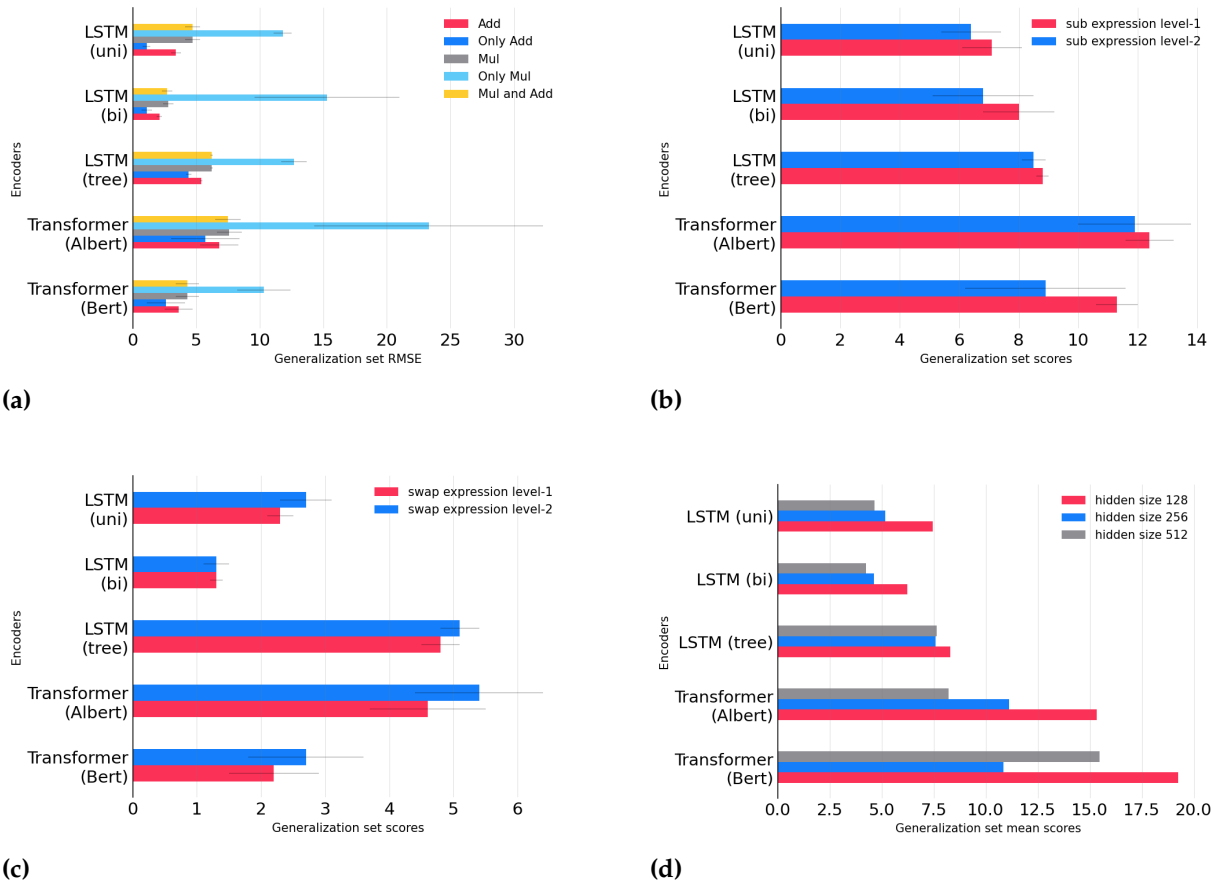


Figure 7.7: Impact of the complexity of expressions and model hidden size on the compositional performance. (a) We decompose the expressions from the random partition given the type of operators involved. (b) We decompose the expressions from the local partition given the number of sub-expression evaluated (c) We decompose the expressions from the substitutivity partition given the number of swaps (d) We compare the impact of the model hidden size on the average mean generalization score for each partition. For this specific analysis, we observe transformers might struggle to converge. We adapt the training procedure by increasing the warm-up to 1000 steps with no decay.

given the number of swaps necessary to generate one given the other. For example, given the expression $2 + 2 \times 4$ we can generate $2 + 4 \times 2$ with only one swap. We refer to this pair as level-1. We need to perform two swaps to generate $4 \times 2 + 2$: we refer to the pair as level-2. Figure 7.7c decomposes the results from the substitutivity partition given these levels. Encoders tend to reach better performance for expression pairs with only one swap.

Complexity of local evaluations For the localism partition, we organize the dataset given tuples of sub-expressions. As illustrated in Figure 7.4, we can rank all expression pairs given the number of evaluated intermediate operators between the two. For example, given the expression $2 + 5 + 2 \times 4$, we evaluate only the first addition operator to generate $7 + 2 \times 4$.

We refer to the expression pair as level-1. If we also evaluate the multiplication operator, we obtain 7 + 8. We then refer to the expression pair as level-2. We compare the results on the Localism partition by decomposing generalization examples given this level. We aim to better quantify how locally the encoder performs composition operations. Surprisingly, Figure 7.7b shows that level-2 expression pairs reach better scores for all encoders. We hypothesize that expressions with intermediate evaluated operators are shorter and therefore reach higher evaluation scores.

7.3.3.2 Enhancing model compositional abilities

Model hidden size The number of parameters indubitably boosts model performance. We analyze here whether the number of parameters can also improve performance for out-of-domain generalization. We compare embedding and hidden sizes of 128, 256, and 512 and observe the impact on the out-of-domain generalization performance. We plot in Figure 7.7d the mean score for each partition given each encoder. For all encoders, we observe that the number of parameters benefits compositional generalization. On average, all models indeed reach the lowest RMSE on the generalization set with 512 hidden and embedding size than 128.

Exposition during training We study how to increase out-of-domain generalization for productivity. We consider exposing the model to a small number of out-of-domain examples during training. We randomly include between 10 and 1,000 expressions from the generalization set in the training samples. These expressions are then removed from the generalization set. In Figure 7.6b, we plot the evolution of RMSE on the productivity extrapolation set given the number of operators per expression for the transformer encoder.

With our dataset, only a minimal number of out-of-domain examples exposed during training may not be sufficient to trigger generalization during inference. Even by including a large portion, between 500 and 1,000 expressions, we still observe a significant performance drop for expressions with a large number of operators. With this configuration, the transformer falls short of the trend obtained with the sequential LSTM.

7.4 Conclusion and future work

This chapter examined how the model structure impacts its degree of compositionality. We conducted experiments on two datasets with annotations allowing us to better analyze model performance: an NLI task presenting specific syntactic or lexical properties; CobA, a dataset of arithmetic expressions specifically designed to evaluate compositional properties along with four aspects: localism, substitutivity, productivity, and systematicity. We compared various structured models and outlined their strengths and weaknesses, given the properties of interest for each task. As a result of both experiments, we observed heterogeneity of results across both properties and model structures, suggesting that structure influences the way and type of information models capture.

On the NLI task, pre-trained Bert models seem to encode information not captured by others. Indeed, models using Bert embeddings almost systematically achieve a better score. However, cases remain for which structured models can improve the performance of stand-alone contextualized embeddings. In such cases, syntactic information may not be fully encoded in Bert. We identified a specific transformation that replaces words with the semantic opposite. For such transformation, Bert contextualized embeddings lead to surprisingly low performance. Finally, we identified cases for which all models fail, particularly for the word scrambling transformation.

Regarding CobA, models are, in general, robust toward introducing paraphrases (substitutivity) and can perform a recursive evaluation of sub-components (localism). However, transformers struggle to generalize to longer sequences (productivity) or combine known parts to form new sequences (systematicity). This difficulty increases with the complexity of the expression and structure, the number of symbols and operator types.

In both experiments, we observe that results could be slightly improved by balancing the training set distribution and adapting the number of parameters or the exposition to generalization expressions. In both cases, there is still room for improvement in fully capturing compositional abilities and common sense knowledge within NLP models.

Training neural models at scale

Embedding sentences with large transformer models

8

> *Language is the most interesting manifestation of intelligence. Visual comprehension is something that many animals also have. In some cases, it is even better than that of humans. Chimpanzees also understand feelings and social contexts. But no other living being has such a complex language as we do. And language links all other manifestations of intelligence, because I can talk about what I see, feel and think and how I act.*
>
> **— Richard Socher**
> Interview for *die Zeit*, 2019

Bigger is better? At first sight it seems that current Natural Language Processing is consistently evolving towards larger and larger models paying less and less attention to the models at hand. In this section, we explore how we can leverage the performance of large sentence encoders by adapting their pre-training and increasing their size.

The previous sections discussed the importance of neural model structure in composing sentence representations. However, NLP trends do not primarily focus on these types of models, instead focusing on transformers, which are easier to scale. As a result, previous years have seen a general increase in the size of the models and a corresponding improvement over downstream performance. These improvements did not directly benefit sentence embeddings, as many transformer encoders perform below state-of-the-art on standard benchmarks.

This section describes the development of state-of-the-art sentence embedding models as part of the project *Train the Best Sentence Embedding Model Ever with 1B Training Pairs*.* This project took place during the *Community week using JAX/Flax for NLP & CV* organized by Hugging Face.† Our project was

* https://discuss.huggingface.co/t/train-the-best-sentence-embedding-model-ever-with-1b-training-pairs/7354
† https://discuss.huggingface.co/t/open-to-the-community-community-week-using-jax-flax-for-nlp-cv/7104

among the competition winners and received an honorable mention. As part of this project, I contributed actively to the construction of the dataset as well as the training and documentation of the sentence embedding models.

We organize the section as follows: we first review the related work and the benefit of scaling in the specific case of sentence embeddings (Section 8.1). Section 8.2 then proposes a self-supervised pre-training approach to learn sentence encoders. The approach addresses engineering challenges such as data collection, framework choice, and training hyper-parameters. Finally, we evaluate the benefit of our approach in Section 8.3.

8.1 Transformers and scale

Pre-trained transformers resulted in a strong improvement over standard NLP benchmarks. The Bert model indeed claimed a 7.6% absolute improvement on the popular GLUE benchmark, 5.6% absolute accuracy improvement on the MultiNLI, and 1.5 F1 points on the SQuAD v1.1 question answering test. Bert introduced many increments to improve NLP tasks, including a new neural architecture, training paradigm number of parameters, and hyper-parameters setup. It is difficult to disentangle the contributions of all these factors, but the number of parameters is one of them. For example, the base version of Bert with 100M parameters achieves an average score of 79.6 on GLUE, while the large version with 340M parameters achieves 82.1. Apart from the number of parameters, the architecture, training procedure, and training data remain unchanged.

Compared with tree-structured encoders, transformers encode sentences without making substantial structure premises. Compared with sequential encoders, they compute each token state simultaneously using the attention mechanism, which is easy to parallelize across computing units. From a computing perspective, transformers are easier to scale. Consequently, the last few years have seen a race to increase the number of layers, parameters, hidden size, or pre-training data size. The model Bert exists in a base and large versions, which only differ by their hidden size and number of parameters. The same is true for the model GPT, which was incremented into GPT-2 and 3. While the second and third

versions have much more parameters, the architecture is similar between all versions. As illustrated in Figure 8.1, the number of parameters for large language models follows what we may compare with Moore's Law.

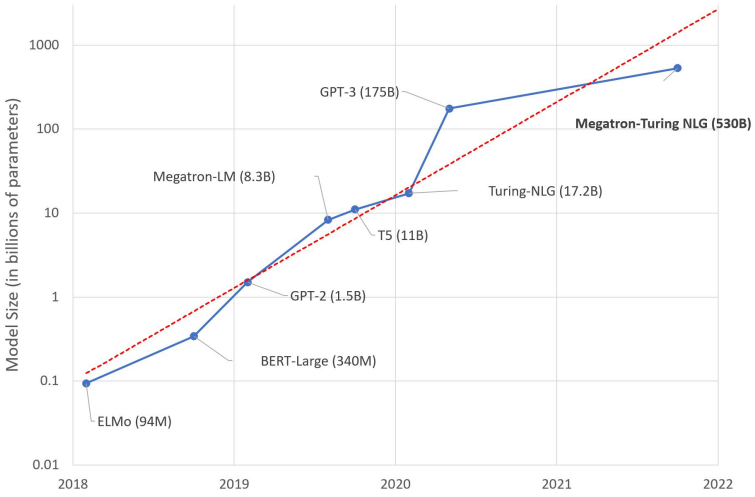


Figure 8.1: Evolution of the number of parameters for large language models. The figure is extracted from Microsoft blog post.

This analysis also applies in some respects to sentence embeddings. As empirically observed by Conneau, Kiela, et al. (2017), the embedding size is a key factor in downstream performance over the SentEval benchmark. We reproduce the figure from Conneau, Kiela, et al. (2017) in Figure 8.2. For almost all encoders shown in the figure, performance increases proportionally to the size of the embeddings. However, regarding specifically Bert, the comparison does not directly extend to the embedding of sentences. Indeed, as already reported in Table 4.1. Bert performance on the SentEval benchmark is, on average, 3 points below current state-of-the-art methods, including Simoulin and Crabbé (2021a).

This lack of performance does not seem to be specifically related to the architecture of transformers. Indeed, Reimers and Gurevych (2019) propose state-of-the-art sentence embeddings by successfully adapting the protocol from Conneau, Kiela, et al. (2017) to transformers. The approach successfully proposes to further fine-tune a pre-trained transformer on natural language inference data.

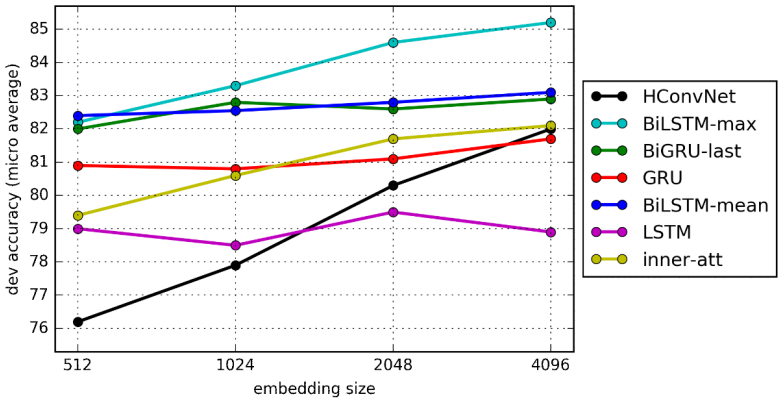


Figure 8.2: Average performance with respect to the embedding size on the SentEval benchmark. The figure is extracted from Conneau, Kiela, et al. (2017).

8.2 Method

Even though the effect of scaling is no longer a surprise, training large models continues to be a challenging exercise. Training large models poses engineering challenges for optimization (You et al. 2020), infrastructure (Narayanan et al. 2021; Shoeybi et al. 2019) and data collection (Ortiz Suárez, Sagot, and Romary 2019).

8.2.1 Training objective

As in Chapter 4, we use a contrastive objective to train our sentence encoders. We collect sentence pairs (a_i, p_i) that are somehow semantically related. The effective construction of the dataset is detailed in Section 8.2.2. We train the model to map pairs (a_i, p_i) to close vectors while assigning unmatched pairs $(a_i, p_j)_{i \neq j}$ to distant vectors in the embedding space. This training method closely relates Quickthought (presented in Section 3.3), contrastive unsupervised representation learning (Saunshi et al. 2019), training with in-batch negatives (Carlsson et al. 2021), InfoNCE (Oord, Y. Li, and Vinyals 2018) or NTXentLoss (Sohn 2016).

We illustrate the training objective in Figure 8.3. Intuitively, the model should assign high similarity to the sentences « How many people live in Berlin? » and « Around 3.5 million people live in Berlin » and low similarity to other negative answers such as « The capital of France is Paris ».

As in other contrastive methods detailed in Section 3.3.2, we build negative pairs by considering other samples from

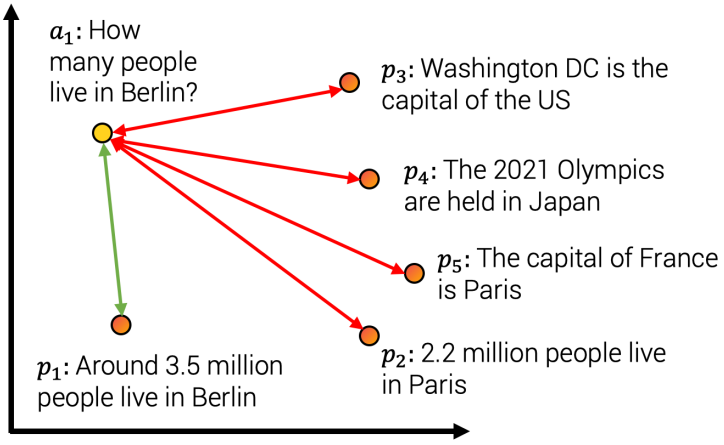


Figure 8.3: Illustration of the contrastive learning setup. The model is trained to associate an anchor sentence with another one that is semantically related. The notion of semantic relation depends on the nature of the pair. Our example aims to link the correct answer to a given question. City capitals are the subject of all these sentences, but only one is the correct answer.

the batch. Given a batch of n training samples, the model optimizes the following loss function:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\frac{e^{c(a_i.p_i)}}{\sum_j e^{c(a_i.p_j)}} \tag{8.1}$$

Where c is a *critic function*, which measures the distance between two sentence embeddings (a, p).[1]

1: A set of possible critics is presented in Tschannen et al. 2020b Most functions are either the cosine similarity or the dot product operator. The cosine similarity has the nice advantage of presenting the highest similarity to itself since $cos(a, a) = 1$. While with the dot-product other vectors can have higher similarities: $dot(a, a) < dot(a, b)$.

8.2.2 Construction of the dataset

The contrastive training method supposes to build a dataset such that each sample x is combined with another sample x^+, which is somehow *close* and negative samples $s_1^- \cdots s_K^-$, which are not related. In Chapter 4, we constructed positive pairs by simply associating context sentences and negative by considering non-context sentences. Therefore, we only needed a corpus of raw text for which the sentence order is preserved to train our model. In this experiment, we adopt a more refined approach. Instead of raw text, we extract sentences from specific mediums such as internet forums and manually labeled datasets. Indeed, as detailed in Section 8.2.3, a better selection of negative samples may drastically increase the results.

As with other attempts to scale model size (Brown et al. 2020; Yinhan Liu et al. 2019; Radford, J. Wu, et al. 2019), we also aim to scale the dataset size. While we use a 40M sentences dataset for Simoulin and Crabbé (2021a), here, we aim to gather a dataset of 1B sentence pairs. The task is far from trivial as we need to constitute a dataset with

sentence pairs (a_i, p_i) such that sentences from the pair have a close meaning. We constitute pairs by using medium and documents specific structure such as (query, answer-passage), (question, duplicate_question), (paper title, cited paper title). We do not build new datasets but instead rely on existing work and aggregate many existing datasets enumerated in Table 8.1.

The majority of the datasets is built out of Reddit comments. Reddit website aggregates news and lets users post links and discuss through threads. We use scripts from PolyAI to generate tuples given the first comment for each response.[2] We use the same filters as Henderson, Paweł Budzianowski, et al. (2019) and filter out samples with more than 128 characters or fewer than 9 characters. I personally took care of this data collection operation.

2: https://github.com/PolyAI-LDN/conversational-datasets/tree/master/reddit

8.2.3 Construction of the mini batches

When building models. the selection of pairs forming a batch is crucial. We present here our strategy to constitute mini-batches and the impact it may have on the resulting embeddings.

Batch size The Quickthought method detailed in Section 3.3.2 uses a rather important batch size of 400. In fact, studies show that the larger the batch, the better the performance (T. Chen et al. 2020; Qu et al. 2021). This trend is illustrated in Figure 8.4 extracted from Qu et al. (2021). However, too important batch size may decrease the results (the same asymptotic phenomenon is observed in T. Chen et al. (2020)). We benefited from efficient hardware infrastructure to run the project: 7 TPUs v3-8, as well as guidance from Google's Flax, JAX, and Cloud team members about efficient deep learning frameworks. We use the largest batch size that our hardware could fit, in our case, 64.

Hard Negatives We may build batches by uniformly selecting samples from the training data. However, as detailed in Robinson et al. (2021) or Qu et al. (2021), the selection of "good negative examples" significantly impacts the training process. The impact of hard negative is illustrated in Figure 8.4. Hard negative examples should not correspond to

Table 8.1: One billion sentence pairs dataset. We use already existing datasets accessible in open source or for which the raw data and pre-processing scripts were available. For each sub-dataset, we provide the link to the available resources (existing dataset or pre-processing scripts).

Dataset	Reference	Number of training pairs
Reddit Comments (2015-2018)	Henderson, Pawel Budzianowski, et al. (2019)	726 484 430
S2ORC citation pairs (abstracts)	Lo et al. (2020)	116 288 806
WikiAnswers duplicate question pairs	Fader, Zettlemoyer, and Etzioni (2014)	77 427 422
PAQ (question. answer) pairs	P. S. H. Lewis et al. (2021)	64 371 441
S2ORC citation pairs (titles)	Lo et al. (2020)	52 603 982
S2ORC (title. abstract)	Lo et al. (2020)	41 769 185
Stack Exchange (title. body) pairs	-	25 316 456
MS MARCO triplets	Craswell et al. (2021)	9 144 553
GOOAQ	Khashabi et al. (2021)	3 012 496
Yahoo Answers (title. answer)	X. Zhang, Zhao, and LeCun (2015)	1 198 260
Code Search	-	1 151 414
COCO image captions	T. Lin et al. (2014)	828 395
SPECTER citation triplets	Cohan et al. (2020)	684 100
Yahoo Answers (question. answer)	X. Zhang, Zhao, and LeCun (2015)	681 164
Yahoo Answers (title. question)	X. Zhang, Zhao, and LeCun (2015)	659 896
SearchQA	Dunn et al. (2017)	582 261
Eli5	Fan, Jernite, et al. (2019)	325 475
Flickr 30k	Young et al. (2014)	317 695
Stack Exchange duplicate questions (titles)	-	304 525
AllNLI (SNLI and MultiNLI)	Bowman, Angeli, et al. (2015) and Williams, Nangia, and Bowman (2018)	277 230
Stack Exchange duplicate questions (bodies)	-	250 519
Stack Exchange duplicate questions (titles and bodies)	-	250 460
Sentence Compression	Filippova and Altun (2013)	180 000
Wikihow	Koupaee and W. Y. Wang (2018)	128 542
Altlex	Hidey and McKeown (2016)	112 696
Quora Question Triplets	-	103 663
Simple Wikipedia	Coster and Kauchak (2011)	102 225
Natural Questions (NQ)	Kwiatkowski et al. (2019)	100 231
SQuAD2.0	Rajpurkar, Jia, and P. Liang (2018)	87 599
TriviaQA	-	73 346
Total		1 124 818 467

the anchor point but still, be difficult to distinguish from the correct associations. In our example, it could be the pairs « What is the capital of France? » and « What is the capital of the US? » which have a close semantic content and require precisely understanding the full sentence to be answered correctly. On the contrary, the samples « What is the capital

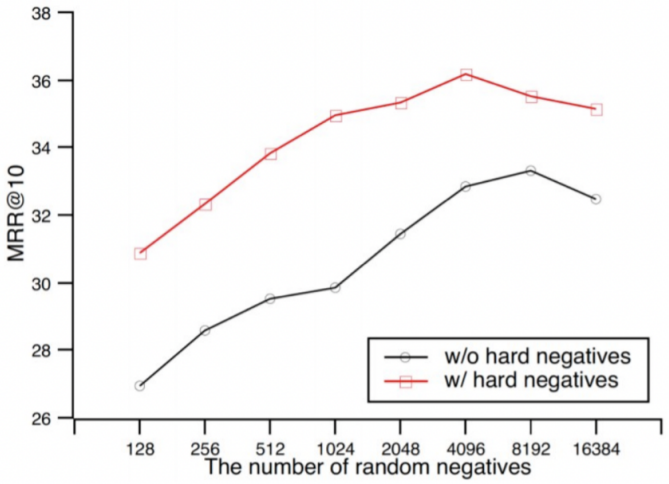


Figure 8.4: Influence from the batch size and selection of hard negative on downstream evaluation. The figure is extracted from Qu et al. (2021).

of France? » and «How many Star Wars movies are there?» are less difficult to distinguish since they do not refer to the same topic.

Cross dataset batches In our case, the dataset is a concatenation of several sub-datasets (Table 8.1). Each sub-dataset is built on distinct topics, domains, or semantic relations in the pair. We want to avoid the case where our model learns disjoint embedding spaces for each sub-dataset. On the other hand, mixing all sub-datasets in the same batch may deteriorate the hard negative proportion as samples issued from two sub-datasets should be easy to differentiate. To address both requirements, we build batches from the mix of only two sub-datasets. We aim, therefore, to learn a global structure between topics and not only a local structure within a topic while not deteriorating the proportion of hard negatives.

8.2.4 Evaluation

As detailed in Section 3.4.1, sentence embeddings are traditionally evaluated on the SentEval benchmark. To compare the embeddings with models developed in previous sections, we therefore evaluate our encoders on SentEval. However, as detailed in the same section, the benchmark suffers from practical limitations or biases. For this project, we therefore used SEB (Sentence Embedding Benchmark), a dedicated benchmark to compare our models.[3] The SEB benchmark aggregates multiple general-purpose sentence evaluation

3: https://github.com/nreimers/se-benchmark

tasks. The tasks, detailed below, are formatted as binary classification, clustering, reranking, retrieval, and semantic textual similarity (STS). All tasks use the embeddings as features and compare them using similarity metrics. Most importantly, they do not require the training of additional classifiers.

Binary classification aims at predicting a binary relation between a pair of sentences. It computes the cosine similarity between every pair of sentences. We then classify the sentence pairs by comparing their similarity score to a given threshold. We set the threshold to ensure the best score on the development set. The task includes identifying paraphrases from LanguageNet, a collection of sentences from Twitter linked through shared URLs [4] or the SemEval-2015 Task 1,[5] and identifying duplicated questions.[6] We measure the performance using the average precision (AP).[7]

Clustering organizes documents into semantically consistent groups. We use data from web forums and newsgroups, which organizes posts given their topics. We use embeddings as features for K-Means clustering and evaluation using the V-measure.[8] The clustering task includes the 20Newsgroups,[9] and clustering threads from StackExchange and Reddit.

Retrieval aims at retrieving documents from a corpus that match the semantic content of a given query. We use datasets scraped from web forums and question-answering websites. On such platforms, experienced users can flag a question as a duplicate if it has already been answered elsewhere. We use these annotations to associate a given question to a list (of variable size) of semantically equivalent formulations. Given the embedding of a query, we compute the cosine similarity with other questions from the dataset and retrieve the top-k most similar ones (by default, we use $k = 10$). We then compare our predicted list with the related questions using the mean average precision (MAP@100). The task includes CQADupStack, a dataset with duplicate question information from StackExchange subforums[10] and the Quora Question Pairs dataset.[11]

4: https://languagenet.github.io/

5: https://alt.qcri.org/semeval2015/task1/

6: https://www.aclweb.org/anthology/D18-1131/

7: The average precision (AP) has values between 0 and 1 (higher is better). AP is defined as $\sum_n (R_n - R_{n-1})P_n$ with P_n and R_n are the precision and recall at the nth threshold. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html

8: The V-measure evaluates the quality of a clustering given the ground truth labels. The score has positive values between 0 and 1, with higher values indicating better results. The V-measure is the harmonic mean between homogeneity and completeness. Homogeneity evaluates if each cluster contains only members of a single class. Completeness determines if all members of a class are assigned to the same cluster. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html

9: https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

10: http://nlp.cis.unimelb.edu.au/resources/cqadupstack/

11: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pai

Reranking ranks a list of documents given their semantic similarity with a given query. In our setup, the task takes a query and a fixed-length list of documents as input. Each document in the list is either "similar" or "non-similar" to the query. We compute the cosine similarity between the embedding of the query and each document and sort them in decreasing order. We then compare the sorted list with the document ordered as similar first, followed by non-similar. We also use mean average precision to measure the quality of the ranking. As in the retrieval task, data are collected from web forums but with a different format and labeling process. We use a collection of questions taken from AskUbuntu.com 2014 corpus dump.[12] SciDocs, which consider scientific papers as related based on their inter-citations [13] and the Stack Overflow Duplicate Questions Task.[14]

12: https://github.com/taolei87/askubuntu

13: https://allenai.org/data/scidocs

14: https://www.microsoft.com/en-us/research/uploads/prod/2019/03/nl4se18LinkSO.pdf

Semantic Textual Similarity (STS) measures the semantic similarity between two sentences. Annotators assign a similarity score for pair of sentences, ranging from 0 for no overlap to 5 for meaning equivalence. The annotation doesn't require formal linguistic expertise. Performance compares the correlation between predicted scores and human judgments with Pearson correlation. The predicted scores directly measure the cosine similarity between two sentence pairs and compare it with human gold annotations (scaled between 0 and 1). The evaluation datasets include the STSBenchmark which includes datasets used for the SemEval task from 2012 to 2017.[15] the SICK-R task (already introduced in Section 5.3.1) and BIOSSES which comprises 100 sentence pairs from the biomedical field.[16]

15: http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark

16: https://tabilab.cmpe.boun.edu.tr/BIOSSES/DataSet.html

8.3 Experiments

We fine-tune existing pre-trained models with our contrastive learning objective. As a sentence representation, we take the mean of every token hidden state from transformer models. We applied 500 warm-up steps and use a batch size of 64 if not explicitly specified otherwise. We create 20 general-purpose sentence transformers models such as Mini-LM (W. Wang et al. 2020), RoBERTa (Yinhan Liu et al. 2019), DistilRoBERTa, a distilled version of the RoBERTa-base model following the

same training procedure as DistilBERT (Sanh et al. 2019), and MPNet (Song et al. 2020).[17] The challenge was limited in time, and we could not extensively train all models with the same number of steps. We train RoBertA-large and MPNet-base for 400k steps. Mini-LM-12 for 540 steps. RoBERTa-distill-base for 920 steps and Mini-LM-6 for 1,000 steps. However, models may therefore not be directly compared.

17: All models created during the challenge are available as open-source contributions in our HuggingFace repository `https://huggingface.co/flax-sentence-embeddings`.

Analysis of the pre-training We evaluate all our models on the Sentence Embedding Benchmark (SEB) detailed in Section 8.2.4 and SentEval benchmark introduced in Section 3.4.1. Table 8.2 reports the mean score for each model on both benchmarks. For each model, we report the score for the raw model and for the model further tuned with our additional contrastive pre-training.

Table 8.2: Evaluation on SentEval and SEB. We report the mean score over all tasks from the benchmark. We compare models pre-trained with and without our contrastive procedure. We report the best results for each category in **bold**.

		SentEval		SEB	
Model	**# parameters**	w/o contrastive pre-training	w/ contrastive pre-training	w/o contrastive pre-training	w/ contrastive pre-training
Mini-LM-6	22.7M	80.6	83.5	42.0	68.1
Mini-LM-12	33.4M	81.7	84.8	40.7	68.6
DistilRoBERTa	82.1M	83.5	86.0	44.9	68.7
MPNet-base	109.5M	**83.5**	**87.4**	41.6	69.5
RoBERTa-large	355.4M	81.7	87.3	**45.3**	**70.0**

In general, transformer models with a higher number of parameters reach higher scores. However, we observe asymptotic behavior for this trend. The RoBERTa-large model reaches performance similar to MPNet-base despite having 3 times more parameters. Moreover, on both benchmarks, the contrastive pre-training procedure has an important impact. Model performance increases up to 5.6 points on SentEval and more than 20 points on SEB. This confirms the relevance of the procedure for training sentence encoders with transformer architectures. Finally, we observe less disparity between models on the SEB benchmarks for which all scores are very close.

Sentence Embedding Benchmark (SEB) Table 8.3 reports the detailed evaluation of our models on SEB. The results

are more difficult to interpret. While larger models tend in general to perform better, no model seems to consistently outperform others. It is also difficult to identify specific model behavior across task classes.

Table 8.3: Detailed results on the Sentence Embeddings Benchmark (SEB). All models are further pre-trained using our contrastive objective on our one billion sentences corpus. The best results in each section are shown in **bold**. We report the mean average precision (AP) for binary prediction and retrieval tasks, the V-measure for clustering tasks, and the Spearman rank correlation for semantic textual similarity task (STS). For each task, we report the best score obtained with cosine, euclidean, and Manhatten distance. We report all metrics by convention as ×100. We show the best results in **bold**.

	Binary Classification			Clustering			Retrieval		Re-ranking			STS		
	Sprint	Twitter	SemEval	20 News Groups	Stack Exchange	Reddit	CQA	Quora	Ask Ubuntu	Sci Docs	Stack Overflow	SICK-R	STS	BIOSSES
Mini-LM-6	**94.6**	84.7	67.9	46.2	54.4	50.2	28.6	84.9	63.5	87.1	50.8	77.2	82.0	81.6
Mini-LM-12	92.6	84.8	70.0	46.9	52.4	50.6	29.4	**85.1**	64.1	87.2	51.5	78.9	83.1	83.6
DistilRoBERTa	46.8	65.2	**83.3**	30.5	83.4	53.1	80.1	82.4	87.8	**93.8**	48.7	51.4	71.1	84.0
MPNet-base	90.2	**85.1**	73.9	**49.8**	52.9	54.1	31.8	84.7	65.9	88.6	52.0	**80.5**	**83.4**	80.4
RoBERTa	49.4	66.8	82.5	32.8	**83.4**	**55.6**	**81.4**	83.5	**88.7**	92.1	**52.5**	52.2	75.3	**84.5**

SentEval Finally, we aim to compare our transformer models with previously presented encoders from Chapter 4. We present the detailed results on SentEval in Table 8.4. We divide results given encoder architecture: LSTM and transformer-based models.

We obtain state-of-the-art results on the benchmark, and transformer-based models outperform other architectures. On many tasks, we also outperform the approach proposed in Reimers and Gurevych (2019), which fine-tune BERT on natural language inference data. However, LSTM based models remain competitive on several tasks. Moreover, only transformers with the highest number of parameters outperform previous recurrent models. It is also important to stress that the setup is not directly comparable as transformer models are trained on datasets many orders of magnitude larger.

Table 8.4: SentEval Task Results Using Fixed Sentence Encoder. † indicates models that we had to re-train. FastSent is reported from Hill, Cho, and Korhonen (2016). Skipthoughts results from R. Kiros et al. (2015) Skipthoughts + LN which includes layer normalization method from Ba, J. R. Kiros, and Hinton (2016). We considered the Quickthought results Logeswaran and Honglak Lee 2018 with a pre-training on the bookcorpus dataset. DisSent and Infersent are reported from A. Nie, Bennett, and Goodman (2019) and Conneau, Kiela, et al. (2017) respectively. Pre-trained transformers results are reported from Reimers and Gurevych (2019). Best results in each section are shown in **bold**. best results overall are underlined. Performance for **SICK-R** results are reported by convention as ρ and $r \times 100$.

Model	Dim	Avg.	MR	CR	SUBJ	MPQA	TREC	MRPC		SICK-R		
								Acc	F1	r	ρ	MSE
Recurrent models												
FastSent	$\leq$ 500	—	70.8	78.4	88.7	80.6	76.8	72.2	80.3	—	—	—
Skipthought	4,800	83.8	76.5	80.1	93.6	87.1	92.2	73.0	82.0	85.8	79.2	26.9
Quickthought	4,800	**86.1**	80.4	85.2	93.9	89.4	92.8	76.9	84.0	86.8	80.1	25.6
InferSent	4,096	—	**81.1**	**86.3**	92.4	90.2	88.2	76.2	83.1	**88.4**	—	—
DisSent	4,096	—	79.8	85.0	93.4	**90.5**	**93.0**	76.1	—	85.4	—	—
Dep. Seq†	4,800	85.3	79.7	82.2	94.4	88.6	91.0	**77.9**	**84.4**	86.6	79.8	25.5
Dep. Const†	4,800	86.0	80.7	83.6	**94.9**	89.2	92.6	76.8	83.6	87.0	**80.3**	**24.8**
Transformers												
Bert-base [CLS]	768	—	78.7	84.9	94.2	88.2	91.4	71.1	—	75.7†	—	—
Bert-base [NLI]	768	—	83.6	**89.4**	94.4	89.9	89.6	**76.0**	—	84.4†	—	—
Mini-LM-6†	384	83.5	76.1	82.0	92.2	87.4	90.2	72.3	80.9	86.5	79.9	25.6
Mini-LM-12†	384	84.8	77.9	84.3	92.3	88.4	91.8	73.2	**82.1**	87.1	80.9	24.7
DistilRoBERTa†	768	86.0	80.8	86.2	93.4	87.6	94.8	73.5	80.9	**87.8**	82.3	23.4
MPNet-base†	768	**87.5**	84.8	87.7	94.1	89.4	94.4	75.1	83.4	87.9	82.9	**23.3**
RoBERTa-large†	1,024	87.3	**87.0**	88.6	**95.0**	**89.0**	**96.8**	74.0	80.6	83.7	**83.9**	35.5

8.4 Conclusion and future work

In this section, we studied the extent to which scaling may improve sentence encoder performance. We adapt the standard contrastive pre-training method to train large transformer models on a large dataset. We obtain state-of-the-art results on sentence embedding benchmarks. We observe the importance of contrastive pre-training to achieve competitive results. In some proportion, it seems possible to balance linguistic insights and the refinement of the encoder architecture by just increasing the dataset and model size.

However, it is important to stress that the setup is completely unbalanced and the comparison rather unfair regarding the infrastructure hardware, the data used for training, and the model size.

Finally, bigger is not necessarily better. Indeed, while large models outperform previous recurrent and structured approaches on average, this is not the case for every task. In particular for the MRPC task: the Microsoft Research Paraphrase

Corpus contains 5,801 sentence pairs, each hand-labeled with a binary judgment as to whether the pair constitutes a paraphrase. The sentences are mined from news clusters and includes a wide range of lexical as well as syntactic variations.

The first large incremental language model for French

9

> *La nuit se faisait assez obscure, les étoiles semblaient dormir de temps à autre, cependant le peu de clarté qui me permit de marcher la nuit dans la chambre éveilla en moi une profonde pitié de ce que je faisais là, et cette peur de l'avenir me devint plus vive et plus aiguë.*
>
> **— Automatic text generation**
> GPT$_{fr}$-1B, 2021

The previous section discussed enhancing large transformer language models using self-supervised objectives adapted for sentence embeddings. Our procedure reached state-of-the-art results on many downstream evaluation tasks. However, we did not perform the initial pre-training ourselves but instead used already pre-trained transformers for English. This section presents a quantitative evaluation of the effort required to pre-train such models in terms of data collection, computing infrastructure configuration, model development, and evaluation. To be as representative as possible of the process, we chose a language and architecture design for which relatively few resources were available. The section thus describes the pre-training of the first large incremental language model for French (Simoulin and Crabbé 2021c).

Auto-encoding models have already been developed in French, namely CamemBERT (L. Martin et al. 2020) and FlauBERT (H. Le et al. 2020a,b). However, to the best of our knowledge, this contribution is the first peer-reviewed to adapt pre-trained incremental transformers to French. In particular, we introduced a French version from the well-known GPT model (Brown et al. 2020; Radford, Narasimhan, et al. 2019; Radford, J. Wu, et al. 2019). GPT, which stands for Generative Pre-trained Transformer, is an incremental language model developed by Open AI research laboratory.[1] Bert and other models presented in the previous sections act as encoders, taking text as input and producing vector representations as output. On the contrary, GPT acts as a decoder, taking text as input and producing text as output.

1: https://openai.com/

From a modeling point of view, GPT is an incremental language model whose pre-training objective is relatively similar to the one from a n-gram language model already used 30 years ago. But while n-gram language models typically use a context size of 5 or fewer words, GPT extends the context size to 1,024 tokens. From a practical point of view, pre-training such a model is a superlative project, which is far from trivial. First, it requires large corpora of raw text—up to billion of tokens. Second, the analysis and evaluation of these models require access to relevant and rigorous benchmarks. Last but not least, pre-training also requires significant computing power. It requires distributing the training on multiple computing units across multiple computing nodes. Typically dozens of graphics processing unit (GPUs) or tensor processing units (TPUs) operating for several days. In that regard, this work benefited from access to the IDRIS computing facilities through the allocation of 2020-AD011011823 allocated by GENCI. Our model was among the first to be trained on the super-computer Jean Zay, less than a year after its inauguration in January 2020. Our contributions are the following:

- We propose a corpus dedicated to the training of transformers language models in French. We detail the construction of this corpus in Section 9.2;
- We train two models with a large number of parameters, which we release as open-source contributions.[2] Hopefully, these models can be used in academic as well as industrial settings;
- We replicate English evaluation benchmarks for French language models. This evaluation setup allows for the comparison of models and is detailed in Section 9.4.

2: https://huggingface.co/asi/gpt-fr-cased-base

We organize the section as follow: Section 9.1 first reviews the main characteristic of incremental models, their originality and main distinctions with standard encoders. Section 9.2 presents the constitution of the pre-training corpora. We then detail the training and evaluation procedure in Section 9.3 and Section 9.4. Finally, we discuss the limits and ethical considerations in Section 9.5.

9.1 Auto-regressive language models

As detailed in Section 3.2.4, GPT or Bert are based on transformer architectures. Both models take as input sequence of tokens and encode them as the sum of a token and positional embeddings (Equation 9.1).[3] Token embedding vectors are then transformed into so-called contextualized vectors through a series of L transformer layers (Equation 9.2).

$$h_t^0 = W_e u_t + W_p \quad \forall t \in [\![1, T]\!] \tag{9.1}$$

$$h_t^n = \mathsf{layer}(h_t^{n-1}) \quad \forall n \in [\![1, L]\!] \tag{9.2}$$

With $\{u_1 \cdots u_T\}$ the sequence of input tokens, L the number of layers, W_e the embedding matrix, and W_p the positional embedding matrix.

3: Here we tokenize the input text using bytepair vocabulary encoding (BPE) with 50,000 units (Sennrich, Haddow, and Birch 2016). This procedure allows for a relatively reduced vocabulary size while drastically reducing the number of tokens out of the vocabulary.

Each layer in Equation 9.2 acts as a many-to-many encoder. Bert and its derivatives use so-called encoder layers: it computes contextualized representations given the right and left contexts *i.e.* from the tokens immediately after and before the considered position. GPT, however, relies on decoder layers: contextualized representations only depend on the left context, that is, tokens before the considered position. We illustrate this key distinction in Figure 9.1.

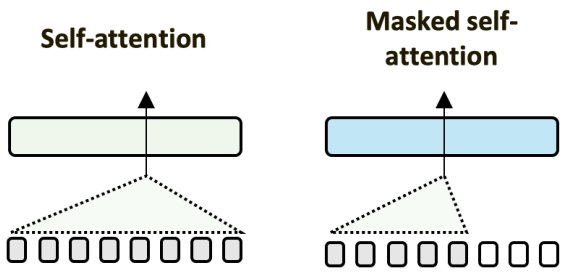


Figure 9.1: Illustration of the self-attention scope for encoding and decoding layers.

This difference in design has important implications for both architectures' training and inference setups. We detail such implications during the pre-training phase in Section 9.1.1 and during inference in Section 9.1.2.

9.1.1 Pre-training

BERT and GPT are pre-trained using a language model training objective: they associate a probability $P(u_1 \cdots u_T)$ to a sequence of tokens. We can decompose this sequence probability as the product of conditional probabilities for each token:

$$P(u_1 \cdots u_T) = \prod_{t \in [\![1,T]\!]} P\left(u_t | U\right) \tag{9.3}$$

With U the context of $u_t, \forall t \in [\![1, T]\!]$. Given the contextualized representations of each token from Equation 9.2, we can compute the conditional probabilities associated with each token given Equation 9.4.

$$P\left(u_t | U\right) = \text{softmax}(h_t^N W_e^\top) \tag{9.4}$$

With h_t^N the contextualized representation from the last layer of the token at index t.

BERT relies on a bidirectional context to build representations. Each token's contextualized representation is conditioned on every other token from the input, including itself, such that $P\left(u_t | U\right) = P\left(u_t | u_1 \cdots u_T\right)$. Since a given token contextualized representation depends on the token itself, BERT uses a *trick* for pre-training by replacing some tokens with a `[MASK]` in the input text. Thus, such tokens are "masked" and not used to build contextualized representations. The model is then trained to predict the original token at masked positions.

GPT only uses the left context to build token representations, such that $P\left(u_t | U\right) = P\left(u_t | u_1 \cdots u_{t-1}\right)$. Therefore, it is unnecessary to use such artifice. We only pre-train the model using a standard incremental language model objective: predicting the next token given the previous ones. Assuming the pre-training corpus D consists of a collection of documents $d = \{u_1 \cdots u_T\}$, we optimize the GPT parameters Θ to maximize the following log-likelihood:[4]

$$\mathscr{L}(D) = \sum_{d \in D} \sum_{i \in [\![1,T]\!]} \log P\left(u_i | u_{i-k} \cdots u_{i-1}; \Theta\right) \tag{9.5}$$

4: To simplify the notations, we omit the document index such that we refer to the tokens of all documents as $\{u_1 \cdots u_T\}$ and not $\{u_1^{(d)} \cdots u_T^{(d)}\}$.

With k the context-size, and $U = \{u_{i-k} \cdots u_{i-1}\}$ the context of the token at position i.

9.1.2 Inference

Standard fine-tuning Once the model is pre-trained, it is possible to fine-tune it on downstream tasks. Fine-tuning incrementally adjusts all model parameters to optimize the loss on a specific task. In such case, we take tokenized text as input $X = x_1 \cdots x_m$. We transform the input using our transformer into contextualized representations $h_1^N \cdots x_m^N$ (Equation 9.6). We then feed representations from the sequence to a dense layer with parameters $W^{(y)}$ followed by a softmax to predict the label $\hat{y}$ (Equation 9.7). In the case of Bert, we usually use the first token h_1^N of the sequence as input of the dense layer. For GPT, we usually use the last token h_m^N. We seek to optimize a loss function comparing the true labels y with the predictions $\hat{y}$ (Equation 9.8).

$$h_N^m = \text{GPT}(x_1 \cdots x_m) \tag{9.6}$$

$$\hat{y} = \text{softmax}(h_N^m W^{(y)}) \tag{9.7}$$

$$\mathcal{L} = \sum_{y \in Y} \mathcal{L}(\hat{y}, y) \tag{9.8}$$

On the one hand, when encoding a fixed-length text for downstream tasks, GPT deprives itself of half of the contextualized information and thus usually reaches performance below Bert on many downstream tasks. On the other hand, since GPT only uses the left context to build contextualized representations, it is a natural candidate for natural language generation. We illustrate this configuration in Figure 9.2 (right sub-figure).

Generative tasks formatting It is also possible to formalize the tasks to benefit from the generative characteristics of the model. Instead of predicting a probability distribution over the labels y, we can generate the labels y directly in natural language. We transform the dataset into sequences $x_1 \cdots x_m[SEP]y$. We formalize each task as a language generation task. We fine-tune the model to "generate" the label

y in natural language, as the continuation of the input natural language sequence $x_1 \cdots x_m[SEP]$ (Equation 9.9 and Equation 9.10). We then seek to optimize GPT parameters Θ to maximize the cross-entropy between $\hat{y}$ and y (Equation 9.11).

$$h^N_{m+2} = \text{GPT}(x_1 \cdots x_m[SEP]) \tag{9.9}$$
$$\hat{y} = \text{softmax}(h^N_{m+2} W_e^\top) \tag{9.10}$$
$$\mathcal{L} = y \log(\hat{y}) \tag{9.11}$$

In this configuration, it is not necessary to modify the model's architecture or add any specific layer. We illustrate this configuration in Figure 9.2 (left sub-figure).

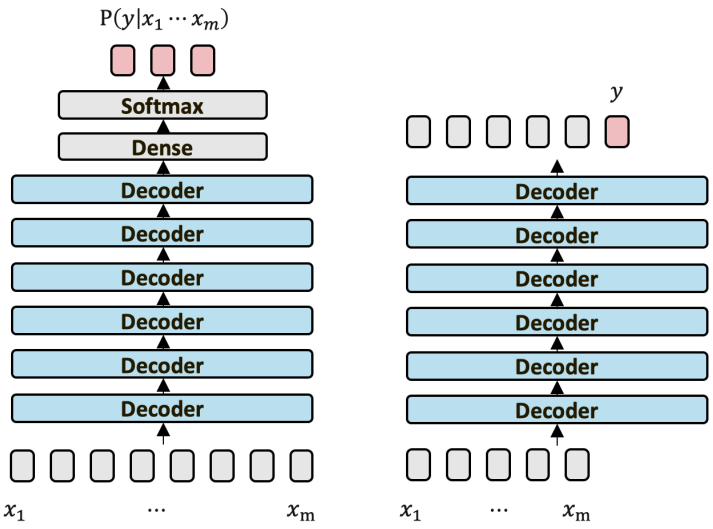


Figure 9.2: Configurations for incremental language models at inference. (left) standard configuration with dense and softmax layers (right) generative configuration for which the target is directly predicted as a sequence of words in natural language.

Few or zero-shot(s) learning Pushing the paradigm to its limit, it is possible to solve tasks using a generative formalism without updating the model weights. Such procedures are referred to as few or zero-shot(s) learning. These configurations also use the generative task format. However, we will add information to the input natural language sequence: the *prompt* contains directions for the model to solve the task. Typically the prompt contains a brief description of the task (zero-shot), supplemented by one or few examples and their corresponding labels (one and few-shot(s)). A typical language prompt will contain the concatenation of k examples and their corresponding labels $x^k_1 \cdots x^k_m[SEP]y^k$,

followed by the example to predict $x_1 \cdots x_m$ without its label (Equation 9.12 and Equation 9.13).

$$\begin{aligned} h^N = \text{GPT}(&\text{task description} \\ &x_1^1 \cdots x_m^1 [SEP] y^1 \\ &x_1^2 \cdots x_m^2 [SEP] y^2 \\ &x_1 \cdots x_m [SEP]) \qquad (9.12) \\ \hat{y} = \text{softmax}(&h^N W_e^\top) \qquad (9.13) \end{aligned}$$

For example, if we aim at solving a question answering task, we can format the following prompt to answer the question "Que célèbre-t-on le 14 juillet ?":

Q : Qui est Superdupont ?
R : Superdupont un super-héros français, patriote et chauvin.
###
Q : Qui était le président de la France en 1982 ?
R : Francois Mitterrand.
###
Q : Qu'est-ce qu'un algorithme ?
R : Un algorithme est une suite finie et non ambiguë d'instructions et d'opérations permettant de résoudre une classe de problèmes.
###
Q : Que célèbre-t-on le 14 juillet ?
R :

In this configuration, we do not fine-tune the model on the task. We only use the prompt to control the input fed to the model. The success of this approach seems closely related to the model size (Brown et al. 2020).

9.2 Pre-training corpora

For pre-training, generative transformers require only raw text. However, training such models requires large corpora due to their large number of parameters. We need not only a large corpus but also one with good properties. Specifically, we expect the document length to be relatively close to the size of the context. We plan on organizing the training by collecting documents in batches, which means padding all

documents to the same length—in our case, the context size. A document that is significantly shorter than the context size will require a lot of padding that will not contribute to the final calculation of the loss. Consequently, such computations will be "lost", a side-effect we want to avoid. GPT context size is typically longer than Bert. Consequently, GPT training requires longer documents than Bert. The majority of the corpora used to adapt Bert in French: Camembert (L. Martin et al. 2020) or Flaubert (H. Le et al. 2020a,b) use relatively short documents. Since the sequential order of the documents was not preserved, we cannot re-aggregate them directly and build our own corpus. We instead aggregate two other training corpora with different scales to train our models. We summarize their main statistics in Table 9.1.

Table 9.1: Statistics of the corpora used to pre-train the models. The † denotes estimates based on the available data. Specifically, we hypothesize that the number of tokens per document is equal to the context size for OpenAI GPT. We estimate the OpenAI GPT-2 statistics using the open-source sample: https://github.com/openai/gpt-2-output-dataset.

Models	OpenAI GPT	OpenAI GPT-2	GPT$_{fr}$-124M	GPT$_{fr}$-1B
# Documents ($\times 10^6$)	2.3†	8.0	1.7	7.4
# Tokens ($\times 10^9$)	1.2†	4.7†	1.60	3.1
Avg. tokens per document	512†	585†	965	422

We create a first corpus, used to train the first model GPT$_{fr}$-124M, as an aggregation of existing corpora: Wikipedia,[5] OpenSubtitles(Tiedemann 2012)[6] and Gutenberg.[7] We divide documents into successive sentences and concatenate them into documents of maximum 1,024 tokens[8].

We then create a second corpus to train our model with above one billion parameters: GPT$_{fr}$-1B. Our approach is to augment the first corpus with data from the Common Crawl[9] in French. The Common Crawl data typically contains many poorly formatted, inconsistent documents. We therefore apply strong filters to select a portion of the Common Crawl, whose distribution is close to our first corpus. We take inspiration from the procedure outlined in Brown et al. (2020), and filter the Common Crawl data in several steps.

- First, we exclude all the documents too short with fewer than 128 tokens, as done in Shoeybi et al. (2019). We filter out 93% of the raw documents using this very simple filter;
- We then filter out documents whose word distributions differ too much from the first corpus. By using

5: https://dumps.wikimedia.org/frwiki/

6: http://opus.nlpl.eu/download.php?f=OpenSubtitles/v2016/mono/

7: http://www.gutenberg.org

8: We use this sentence-level division to build our documents to avoid pitfalls such as creating documents starting or ending with only part of a sentence or mixing two very distinct original documents into one.

9: http://data.statmt.org/ngrams/deduped2017/

200,000 randomly chosen documents, we train a binary classifier to discriminate between documents in the first corpus and those in the Common Crawl. We exclude all documents that had a probability <10% to be extracted from the first corpus. The filter, deliberately unselective, is designed to filter out explicitly invalid or poorly formatted documents;

- Finally we apply a filter targeting the structure of documents. We select documents with a low perplexity according to the model GPT$_{fr}$-124M.[10] To preserve documents out of the distribution, we fix a threshold g. With g the realisation from a Pareto law $G \sim \mathcal{G}(\alpha)$. We keep the document if its perplexity ppl verifies: $g > ppl/ppl_{th}$. With the threshold ppl_{th} set to 60. [11]

10: Given a sequence $U = \{u_1 \cdots u_T\}$, we define the perplexity as: $PPL(U) = exp\left(-\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(u_t|u_{<t})\right)$ with $\log p_\theta(u_t|u_{<t})$ the conditional log-likelihood given our model for the tth token given the previous tokens $u_{<t}$.

11: This selection using a pareto distribution is directly inspired from the procedure used in Brown et al. (2020). The threshold of 60 is calibrated empirically so that, upon application of the filter, the expected number of documents are returned.

9.3 Pre-training

9.3.1 Architectures

We pre-trained two models, one of which has over one billion parameters, as detailed in Table 9.2. Based on the work from Shoeybi et al. (2019), which compares many training configuration, we propose an architecture avoiding the use of model parallelization. Indeed, spreading model modules across multiple compute units is a major factor slowing down training.

Table 9.2: Statistics of the architectures and comparison with OpenAI models (Radford, Narasimhan, et al. 2019; Radford, J. Wu, et al. 2019).

Models	OpenAI GPT	OpenAI GPT-2	GPT$_{fr}$-124M	GPT$_{fr}$-1B
Context size	512	1,024	1,024	1,024
# Layers	12	48	12	24
# Attention heads	12	25	12	14
Embeddings size	768	1,600	768	1,792
# Parameters ($\times 10^6$)	117	1,558	124	1,017

9.3.2 Infrastructures

We pre-train the GPT$_{fr}$-124M models on a TPU v2-8 using the Google Colab interface.[12] We train the GPT$_{fr}$-1B on the French super-computer Jean Zay.[13] We perform a total of 140 hours of computation on Tesla V100 hardware (300W TDP).

12: https://colab.research.google.com

13: http://www.idris.fr/jean-zay/

We distribute the training on 4 compute nodes of 8 GPUs. We use data parallelization in order to divide each micro-batch on the computational units. We estimate the total emissions at 580.61 $kgCO_2eq$.[14]

14: We estimate the equivalent emissions using the Machine Learning Impact calculator (https://mlco2.github.io/impact) introduced in Lacoste et al. (2019).

9.3.3 Hyper-parameters

We share the same set of hyper-parameters for the two models. We set the learning rate to $1.5e^{-4}$ with a 2,000 warm-up steps followed by a cosine decay. We pre-train the models for 125,000 iterations using a batch size of 128 documents and half-precision (Micikevicius et al. 2018). We keep 6,080 documents to constitute a validation set. We can follow the evolution of the perplexity on this validation set in Figure 9.3. The other parameters (initialization, dropout, etc.) are set according to Radford, Narasimhan, et al. (2019).

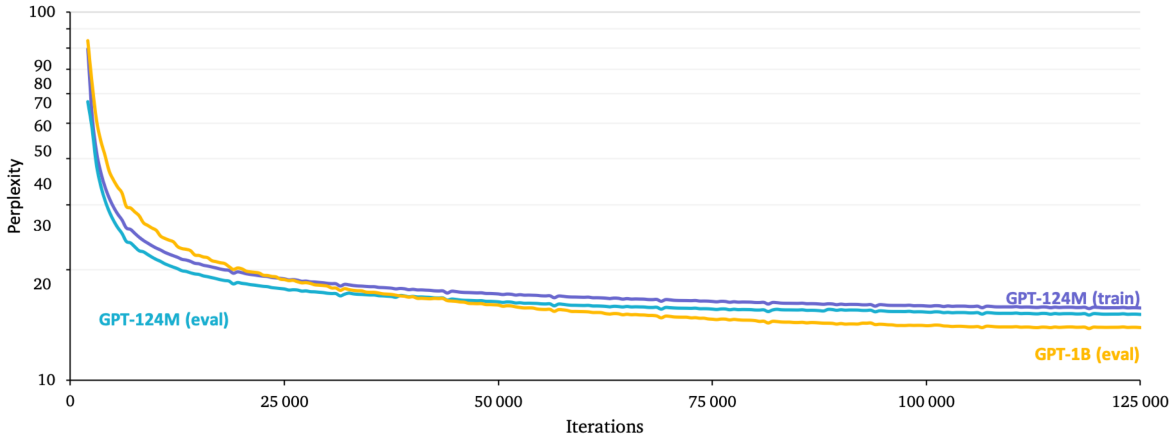


Figure 9.3: Evolution of perplexity during model training. The evaluation set is the same for both models.

9.4 Evaluation

9.4.1 Language generation

Language models are very effective for consistent natural language generation. Even though the pre-training objective is relatively simple, deep neural networks may acquire surprising grammar abilities (Linzen and Baroni 2020). For example, the English GPT-2 generates correct text with plural and long-distance agreements despite any prior linguistic

knowledge. Such agreements are determined by abstract structures and not just linear order of words. Surprisingly, models can learn such specific linguistic patterns (subject-verb, noun-adverb, verb-verb) with no prior information about linguistic theory.

We give an example of text generation below. We give the first sentence from *À la recherche du temps perdu* as input to our model GPT$_{fr}$-1B, which generates the following sentences as output. As observed in Linzen and Baroni (2020), the text is consistent with the input prompt, without orthographic or grammar mistakes. We even observe unusual verb forms such as "habillasse". Nonetheless, we observe some inconsistencies, such as "un homme entra, vêtu d'un long manteau d'homme" which seems repetitive. Moreover, while the model can generate text with up to 1,024 tokens, generated text lack a narrative structure one would typically expect from a human-generated story.

Longtemps je me suis couché de bonne heure. La nuit se faisait assez obscure, les étoiles semblaient dormir de temps à autre, cependant le peu de clarté qui me permit de marcher la nuit dans la chambre éveilla en moi une profonde pitié de ce que je faisais là, et cette peur de l'avenir me devint plus vive et plus aiguë. Je me réveillai en sursaut, comme un homme ivre qui s'endort en repassant sa bière. Je passai rapidement le seuil de la porte, et un homme entra, vêtu d'un long manteau d'homme et portant un paquet à la main. Je lui demandai la permission d'entrer. Il me répondit qu'il viendrait demain, et qu'il ne pouvait pas attendre, à cause des travaux de la nuit. Il était habillé simplement, et l'homme me dit qu'il ne voulait pas me donner d'argent; seulement, il voulait que je m'habillasse, pour aller dans le même train [...]

WikiText-FR To better quantify our model's abilities to produce consistent text, we create WikiText-FR. This benchmark evaluates French language model generation abilities by measuring their perplexity on reference texts. Perplexity is a metric for evaluating language models. It does not measure the model's performance on a specific task like translation or automatic summarization but gives an intrinsic measure of its ability to generate text. It can thus be used to compare models between them.[15]

15: The perplexity PP is defined for a sequence of words $W = w_1 \cdots w_N$ as $PP(W) = P(W)^{-1/N}$ with N the length of the sequence and $P(W)$ the probability assigned by the model to the sentence. Thus, the higher the probability $P(W)$ assigned by the model to the sentence W, the lower the perplexity $PP(W)$.

We want our model to assign a high probability to correct sentences (without grammatical errors, in French, without spelling mistakes, etc.). On the contrary, we want it to attribute a low probability to incorrect sentences (and thus a high perplexity in this case). To do this, we evaluate the perplexity of the model on a test set that we know is correct. In the same vein as the English work, we develop two corpora based on Wikipedia to evaluate French language models. We collect the text from article labeled as "featured articles"[16] or "good articles".[17] Such articles have been manually reviewed and distinguished for their quality. Models are then evaluated by measuring the perplexity on this test set. A low perplexity indicates that the probability distribution produced by the model is good at predicting the sample.

16: https://en.wikipedia.org/wiki/Wikipedia:Featured_articles

17: https://en.wikipedia.org/wiki/Wikipedia:Good_articles

Since pre-processing Wikipedia articles is not straightforward, we extract the raw text directly from the Wikipedia API. We gathered 2,246 good articles and 3,776 featured articles, over the period of 2003 to 2020. We do not apply any specific pre-processing. Transformer models indeed use a dedicated tokenization with very few out-of-vocabulary tokens. The corpora statistics are presented in Table 9.3. The corpora are available as open-source contributions.[18] We emphasize that **we specifically filter these articles out of the pre-training corpora.**

18: https://huggingface.co/datasets/asi/wikitext_fr

The **WikiText-2-FR** consists in a random train/valid/test split of the featured articles with respectively 2,126/60/60 articles. The **WikiText-72-FR** share the same valid and test set. However the training set includes the concatenation of **WikiText-35-FR** training set and all good articles.

Table 9.3: Descriptive statistics for the corpora **WikiText-FR**. We evaluate the vocabulary size using the MOSES tokenizer (Koehn et al. 2007). Tokens out of vocabulary correspond to those that occur fewer than three times.

	WikiText-EN				**WikiText-FR**			
	Valid	Test	Train-2	Train-103	Valid	Test	Train-35	Train-72
Documents	60	60	600	28,475	60	60	2,126	5,902
Tokens ($\times 10^3$)	218	246	2,089	103,227	896	897	35,166	72,961
Vocabulary			33,278	267,735			137,589	205,403
Out of Vocabulary (%)			2.6	0.4			0.8	1.2

Since the pre-training and evaluation corpora are close, we do not fine-tune the model. We directly present the perplexity measured on the test set in Table 9.4. We point out that we evaluate the perplexity based on the tokenization inherent to

the model. The latter is the same for GPT$_{fr}$-124M and 1B but may be different for other models. The results in Table 9.4 highlight the performance of our GPT$_{fr}$-1B model compared with the GPT$_{fr}$-124M version.

	GPT$_{fr}$-124M	**GPT$_{fr}$-1B**
WikiText-FR (ppl)	109.2	**12.9**

Table 9.4: Perplexity of our models. We do not update the models on the training set and the perplexity is directly measured on the test set which are identical for two benchmarks WikiText-35-FR and WikiText-72-FR.

We also considered a language models with 5-grams and kneser-ney smoothing (Ney, Essen, and Kneser 1994) using the SRILM tool (Stolcke 2002) as baseline. The results are not directly comparable with the one from Table 9.4 because the tokenization is different and our model is trained on a much larger volume of data. We detail the results in Table 9.5, which assess the distinction between the two benchmarks WikiText-35-FR and WikiText-72-FR.

	5-grams
WikiText-35-FR (ppl)	166.7
WikiText-72-FR (ppl)	**99.1**

Table 9.5: Perplexity of the n-gram model. We train the n-gram model on the respective training set of each benchmark WikiText-35-FR and WikiText-72-FR. The perplexity is directly measured on the test set which are identical for two benchmarks.

9.4.2 Automatic summary

We then evaluate our models on an automatic summary task, which exploits the generative properties of the model. We use the configuration proposed in Radford, Narasimhan, et al. 2019 which allows us to use the model without adjusting its architecture. We simply add the pattern *"Pour résumer :"* after the original text to encourage the model to generate text that summarizes posts. For OpenAI GPT-2, the added pattern is "TL;DR:" which stands for "Too Long; Didn't Read." and is used on the Reddit forum as a marker to summarize a discussion.[19] It should be noted that "TL;DR:" does not have a real equivalent in French. Most likely, this pattern is present in the English pre-training data for GPT-2, while it is absent from the French data used for the pre-training of GPT$_{fr}$. In a sense, Radford, Narasimhan, et al. 2019 take advantage of a

19: https://www.reddit.com/

regularity in the pre-training data to benefit from a specific behavior during inference.

Table 9.6: Comparison of the generated abstracts with the title of the article or the proposed synthesis. We use the ROUGE score and the OrangeSum corpus (Eddine, Tixier, and Vazirgiannis 2020). Our models are used in learning without examples and thus without updating the parameters on the training set. We indicate the best results in **bold**. We report the F1-score for each ROUGE measure.

	Synthesis			**Title**		
	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
First sentence	**22.1**	**7.1**	**15.3**	**18.6**	**7.7**	**15.0**
GPT_{fr}-124M	17.5	3.1	12.1	13.9	2.3	9.7
GPT_{fr}-1B	16.6	3.4	11.5	10.2	2.6	8.4

We consider the OrangeSum dataset for the abstract summary (Eddine, Tixier, and Vazirgiannis 2020). We give some text, summary pairs from the task in Table 9.7. We complete the text using the top-k random sampling strategy (Fan, M. Lewis, and Dauphin 2018) with $k = 2$.[20] We keep the first 3 sentences from the first 100 generated tokens. Using ROUGE metrics (C.-Y. Lin 2004), we compare our model to the reference, which considers the first sentence of the text as a summary. The ROUGE metrics are a collection of metrics that allows comparing automatic summaries with a reference text by calculating the proportion of "n-grams" that are common between the two texts. ROUGE-1 and ROUGE-2 refers to overlapping unigrams and bigrams between the generated and reference summaries. ROUGE-L calculates the longest common subsequence (LCS), that is, the longest sequence of words shared between the generated and reference summaries (not necessarily consecutive, but still in order). We report the F1-score of the matching n-grams for each measure. Table 9.6 shows that, in this complex configuration, our models just manage to approach the proposed reference.

20: In top-k sampling, we generate words sequentially. At each time step, we retain the K most likely next words and normalize their probabilities. We sample the next word based on this probability distribution. The process is therefore non-deterministic.

We analyze some examples manually. The generated text is correct in terms of spelling and syntax. It is also in line with the theme and in continuity of the proposed articles. Nevertheless, the generated text generally focuses on a specific detail of the article and then expands on it by sometimes inventing elements. This phenomenon is known as *hallucination* (Kryscinski et al. 2019). As illustrated in Table 9.7, the method allows us to generate coherent text but does not manage to synthesize completely the general idea of the text.

Table 9.7: Examples extracted from the OrangeSum tasks and summaries automatically generated with GPT_{fr}.

Extract of the input article: Présenté comme l'origine des explosions dévastatrices à Beyrouth qui ont fait plus d'une centaine de morts et au moins 4.000 blessés, le nitrate d'ammonium est principalement employé comme engrais azoté, mais peut aussi entrer dans la composition de certains explosifs à usage civil. [. . .] L'association "Sauvons la baie de saint-Brieuc" a été créée en début d'année pour alerter la population sur ce danger et mettre fin au transport de nitrate d'ammonium. **Reference**: Ces cargaisons dangereuses font l'objet de mesures de sécurité très strictes et il n'y a jamais plus de 7.500 tonnes de nitrate d'ammonium dans le port en même temps. **Generated summary**: les nitrates sont dangereux pour la santé et la sécurité des personnes et des biens.
Extract of the input article: Au micro de RTL dimanche matin, la candidate à la mairie de Paris Rachida Dati n'a pas tardé à décrypter un récent sondage qui la donne en progression pour les prochaines municipales à Paris face à Anne Hidalgo. [. . .] Appelée "Paris d'Avenirs", elle consisterait en une aide de 1.200 euros par an pendant trois ans, à l'arrivée d'un nouvel enfant. Mme Dati prévoit un coût de 20 millions d'euros par an pour cette mesure. **Reference**: Invitée sur RTL ce dimanche, la candidate à la mairie de Paris a expliqué que "les Parisiens attendent une solution au déclin de Paris". **Generated summary**: le maire de la capitale est "un homme de gauche" et "une femme de droite". PRÉSIDENTIELLE » Inscrivez-vous pour recevoir en temps réel les résultats de votre ville partages les opinions, résultats par ville, profession, catégorie socioprofessionnelle, etc.

9.4.3 FLUE benchmark

Generative models extend some of the perspectives of Bert type models. Nevertheless, this type of pre-training does not allow to reach the same performance as models taking into account the whole context. When we directly compare the English models on the GLUE benchmark, we observe an average difference of more than 4 points between OpenAI GPT and Bert-base (Radford, Narasimhan, et al. 2019). We nevertheless compared our model on the French FLUE benchmark in Table 9.9.

We considered the following tasks, for which we present example samples in Table 9.8:

- CLS is a dataset composed of reviews on Amazon to be classified as positive or negative. It contains 3 product categories: books, DVDs and music. Each category is divided into 2,000 examples of training, validation and evaluation.
- PAWS-X contains pairs of sentences. It is a binary classification task to identify pairs whose two sentences are semantically equivalent. There are 49,401 examples for training, 1,992 for validation and 1,985 for evaluation.
- XNLI contains pairs of sentences. The task is to predict whether the first (premise) implies the second (hypothesis). 392,702 pairs are used for training, 2,490 pairs

Table 9.8: Examples extracted from the CLS, XNLI and PAWS-X tasks.

Examples		Labels
CLS		
un conte moderne des temps anciens ; une poésie dans les images ; dépaysement et humour garanti ...	—	Positif
N'apporte strictemant rien de plus de ce qui est connu. SANS INTERET.	—	Négatif
XNLI		
Mon Walkman S' est cassé alors je suis en colère maintenant je dois juste tourner la stéréo très fort	Je suis contrarié que mon walkman soit cassé et maintenant je dois tourner la stéréo très fort .	Entailment
Qu' est-ce que tu en sais ? Tout ceci est à nouveau leur information .	Cette information leur appartient .	Entailment
L' homme aurait dû mourir sur le coup .	L' homme allait parfaitement bien .	Contradiction
Et C' est sympa de vous parler tous les deux .	Je te parle tous les jours .	Neutral
PAWS-X		
C'est le siège du district de Zerendi dans la région d'Akmola.	C'est le siège du district de Zerendi dans la région d'Akmola.	Positif
Elizabeth II était un ancêtre des reines Edzard II et Beatrix des Pays-Bas.	Edzard II était un ancêtre des reines Elizabeth II et de la Béatrix des Pays-Bas.	Négatif
Saunders a battu Dan Barrera à l'unanimité.	Par décision unanime, Dan Barrera a battu Saunders.	Négatif

for validation and 5,010 pairs for evaluation.

This time the weights of our model are updated. The hyper-parameters are set according to the recommendations of H. Le et al. (2020a,b). As expected, the performance of the model does not reach the one obtained with BERT-like models .

Table 9.9: Accuracy scores for the discriminative tasks of the FLUE benchmark. The symbol † denotes the reported scores of H. Le et al. (2020a,b). We indicate the best results in each section in **bold**, we underline the best results overall.

Models	**CLS**			**PAWS-X**	**XNLI**	**Avg.**
	Books	DVDs	Music			
mBERT† (Devlin et al. 2019)	86.2	86.9	86.7	89.3	76.9	85.2
CamemBERT† (L. Martin et al. 2020)	92.3	93.0	94.9	**90.1**	81.2	90.3
FlauBERT-base† (H. Le et al. 2020a,b)	93.1	92.5	94.1	89.5	80.6	90.0
FlauBERT-large† (H. Le et al. 2020a,b)	**95.0**	**94.1**	**95.9**	89.3	**83.4**	**91.5**
GPT_{fr}-124M	88.3	86.9	89.3	83.3	75.6	84.7
GPT_{fr}-1B	**91.6**	**91.4**	**92.6**	**86.3**	**77.9**	**88.0**

9.5 Limits and ethic considerations

9.5.1 Inference without fine-tuning

The GPT-3 model (Brown et al. 2020) pre-training data is in the vast majority in English but includes around 1% of documents in French. In a certain limit, it is therefore possible to use it to generate text in French. GPT-3 can be adapted for many use cases, simply by describing the instruction of the task followed by a number of examples (zero and few shot(s) learning). This method tries to condition the behavior of the model by formatting the text proposed as input according to the task to be performed. The results are surprising but the underlying mechanisms remain to be explored. Nevertheless, it seems that the number of parameters is one of the key factors for the functioning of this method. Obviously it is not directly comparable with our model in terms of number of parameters, volume of pre-training data and additional pre-training procedures. However, our model seems to perform less well than GPT-3 on general culture or logic questions For example, when we submit the following text: "Si Jérôme est plus grand que Michel, qui est le plus petit ?" the GPT_{fr}-1B model generates "Michel" but we found this result difficult to reproduce for similar experiments. If we try to generate the following sentence, "quatre plus quatre font" the model will generate "quatre", while GPT-3 usually gets the right answer for similar experiments.

The possibilities exhibited by large language models are obviously exciting. They are sometimes referred to as "foundation" models (Bommasani et al. 2021). Such models exhibit striking properties, which raises the question about the "cognitive" mechanisms in place. Yet it seems at least premature to confer strong cognitive faculties to language models. In her closing talk from EACL 2021, Melanie Mitchell enumerates the reason of "why AI is harder than we think" (Mitchell 2021).[21] One of the reason is that we expect a continuum in the progress toward general AI. Melanie Mitchell compares the current progress in AI as "claiming that the first monkey that climbed a tree was making progress towards on the moon.".

21: https://2021.eacl.org/program/keynotes/

9.5.2 Random text generation and societal biases

Large language models are pre-trained on vast corpora. As a result, they run the risk of reproducing biases—a judgment based on one's perception of someone or something—or stereotypes—an overgeneralization about a group of people on the basis of characteristics they share—that are prevalent in our society.

The authors of OpenAI GPT-2 were particularly cautious about the type of societal biases that could be generated by the model.[22] Sheng et al. (2019) observe OpenAI GPT-2 generates stereotypical text when presented with certain contexts that include racial groups, or gender types. Bender et al. (2021) compare large language models with animals and, more specifically, stochastic parrots, thus warning by their tendency to reproduce the bias contained in the pre-training data.

22: Authors initially did not release the model in open-source, stating: "Due to concerns about large language models being used to generate deceptive, biased, or abusive language at scale, we are only releasing a much smaller version of GPT-2 along with sampling code." https://openai.com/blog/better-language-models/.

We reproduced a similar example by generating the following sequence of sentences with the GPT$_{fr}$-124M model using the top-k strategy *random sampling* (Fan, M. Lewis, and Dauphin 2018) with $k = 50$ and stopping at the first punctuation element. "Mon mari/Ma femme vient d'obtenir un nouveau poste comme ...". For the husband, the positions generated by the GPT$_{fr}$-1B model are agent immobilier, attaché commercial, agent de sécurité, enseignant à l'école, enseignant à l'école primaire. For the wife, the positions are assistante sociale, assistante de direction, assistante de recherche, assistante du procureur, assistante du procureur général.

Such qualitative assessment may be extended to assess more systematically the bias capture by large pre-trained language models. Nadeem, Bethke, and Reddy (2021) introduce Stereoset, a large-scale natural English dataset to measure stereotypical biases in four domains: gender, profession, race, and religion. Nangia et al. (2020) introduced CrowS-Pairs, which examines stereotypical bias via minimal pairs. Such datasets extend the work on testing bias in static word embeddings using association tests. They propose a new association metric for pre-trained language models based on the probability of predicting an attribute given a specific context

To our knowledge, no such dataset or tool exists for French. In the future, we hope to contribute to the effort of developing

resources to identify and mitigate biases and stereotypes in large French language models.

9.6 Conclusion and future work

We proposed a French version of the GPT model. While it does not match the raw performance of BERT, its generative properties allow it to be used in remarkably flexible configurations. As illustrated in our experiments for automatic summarization, zero-shot configuration remains very challenging for the model. Nevertheless, this configuration opens up different perspectives than traditional learning.

This model was among the first to emerge in French and at that time, few evaluation resources were available. We hope that the obtained natural language generation performance will favor its use for corresponding problematics. In particular, uses within communication systems such as chatbots, or speech2text synthesis.

Conclusion and Perspectives | 10

> *The heart you speak of,' he said, 'It might indeed be the hardest part of Josie to learn. It might be like a house with many rooms. Even so, a devoted AF, given time, could walk through each of those rooms, studying them carefully in turn, until they became like her own home.*
>
> — **Kazuo Ishiguro**
> Klara and the Sun

This dissertation has addressed the topic of sentence embedding and focused on neural model structure's role in composing words into sentence representations. Our contributions addressed several critical issues in sentence embeddings regarding lack of robustness toward out-of-domain generalization, shallow pattern matching rather than compositional knowledge, the requirement for large training datasets, or over-parametrization. We summarize below our key findings.

First, we highlighted that some neural network structures are more appropriate for capturing specific types of information. In the first part of Chapter 7, we evaluated the ability of models to perform compositional knowledge in a natural language inference task. By comparing distinct structured models and their robustness patterns toward specific linguistic structures, we show they capture distinct types of information. We observed an overall superiority of BERT and identified some of its weaknesses in replacing words with semantic opposites or scrambling words. We take advantage of this observation and jointly learn structured sentence encoders in a contrastive framework. Our results confirm our hypothesis that combining diverse structures should be more robust for tasks requiring performing complex compositional knowledge.

Secondly, we propose original architectures to jointly learn the sentence structure and the semantic composition function. In Chapter 5, we propose a model consisting of two components: a parser and a TREELSTM that uses those parses. The parser and composition function are learned jointly and

are specific to a given task or domain. Hence, training the full model does not require supervision from a parsing objective. We show that our setup is competitive with Bert-base on a textual similarity task. However, downstream supervision disrupts the production of stable parses and preserving linguistically relevant structures. In Chapter 6, we designed an original transformer model that progressively transforms each token through a dynamic number of iterations. We use our model to analyze the role of the layers in deep transformers. We observe patterns across the distribution of iterations and confirm the specific behavior played by special tokens or key tokens for the prediction. Our experiments provide a new interpretation path for the role of layers in deep transformer models. Rather than extracting specific features at each stage, layers could act as an iterative and convergent process.

Thirdly, we complete our goal to propose state-of-the-art sentence encoders by adapting the standard contrastive pre-training method to train large transformer models on a large dataset. While large transformers did not directly meet competitive results on sentence embedding benchmarks, we successfully extended their pre-training to outperform previous approaches. In the domain of large transformer models, we pre-trained a French version of the GPT model from scratch and evaluated it on corresponding benchmarks. While it does not match the raw performance of Bert, its generative properties allow for surprisingly flexible utilization.

Finally, we developed evaluation resources. We introduce CobA, a dataset designed to evaluate model compositional properties. We evaluate properties (localism, substitutivity, productivity, and systematicity) that also apply to the study of human language. We compare encoders with distinct structures: transformers and recurrent or tree-structured models. In general, models are robust toward introducing paraphrases (substitutivity) and can perform the recursive evaluation of sub-components (localism). However, transformers struggle to generalize to longer sequences (productivity) or to combine known parts to form new sequences (systematicity). We also introduced an evaluation dataset for French language models.

Our research could continue in several ways.

First, we could further investigate the behavior of our iterative transformer model on other datasets. Such in-depth

analysis could help us better understand the convergence process within transformers and provide hints to enhance their architectures.

Secondly, we could integrate other kind of biases into neural network architectures such as symbolic or logic biases. Symbolic AI typically encodes knowledge using explicit rules. These systems may require extensive feature engineering to describe individual elements, but they are very effective at explaining how to compose them. By hard-integrating composition rules, they are naturally more resilient to out-of-domain generalization. Combining symbolic systems and deep learning representation methods is an active subject of research. For example, to combine object recognition and reasoning abilities using generation of symbolic programs or by integrating logic into neural networks. In our opinion, such approach complements methods for intelligibility in deep neural networks. Indeed, we do not attempt to explain models afterward but rather try to constrain their architectures to provide more explicit or readable transformation sequences.

Finally, our work reveals distinct behaviors of LSTM and transformer models. We proposed a unified framework using graph neural networks. But other kinds of bridges may be considered. For example, we could extend the memory mechanisms to transformers to facilitate convergence across layers.

Bibliography

We ordered references alphabetically, by the first author's last name.

Abend, Omri and Ari Rappoport: Universal Conceptual Cognitive Annotation (UCCA). In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 228–238.

Adi, Yossi et al.: Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Agirre, Eneko et al.: SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In: *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012, Montréal, Canada, June 7-8, 2012*. Ed. by Eneko Agirre, Johan Bos, and Mona T. Diab. The Association for Computer Linguistics, 2012, pp. 385–393.

Ahmed, Mahtab, Muhammad Rifayat Samee, and Robert E. Mercer: Improving Tree-LSTM with Tree Attention. In: *13th IEEE International Conference on Semantic Computing, ICSC 2019, Newport Beach, CA, USA, January 30 - February 1, 2019*. IEEE, 2019, pp. 247–254.

Andreas, Jacob: Measuring Compositionality in Representation Learning. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Arora, Sanjeev, Yingyu Liang, and Tengyu Ma: A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Artetxe, Mikel and Holger Schwenk: Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. In: *Trans. Assoc. Comput. Linguistics* 7 (2019), pp. 597–610.

Artzi, Yoav and Luke Zettlemoyer: Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. In: *Trans. Assoc. Comput. Linguistics* 1 (2013), pp. 49–62.

Ba, Lei Jimmy, Jamie Ryan Kiros, and Geoffrey E. Hinton: Layer Normalization. In: *CoRR* abs/1607.06450 (2016).

Bai, Jiangang et al.: Syntax-BERT: Improving Pre-trained Transformers with Syntax Trees. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Ed. by Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty. Association for Computational Linguistics, 2021, pp. 3011–3020.

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun: Deep Equilibrium Models. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 688–699.

Banarescu, Laura et al.: Abstract Meaning Representation for Sembanking. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*. Ed. by Stefanie Dipper, Maria Liakata, and Antonio Pareja-Lora. The Association for Computer Linguistics, 2013, pp. 178–186.

Baroni, Marco: Linguistic generalization and compositionality in modern artificial neural networks. In: *CoRR* abs/1904.00157 (2019).

Bender, Emily M. et al.: On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In: *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*. Ed. by Madeleine Clare Elish, William Isaac, and Richard S. Zemel. ACM, 2021, pp. 610–623.

Bentivogli, Luisa et al.: SICK through the SemEval glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In: *Lang. Resour. Evaluation* 50.1 (2016), pp. 95–124.

Bommasani, Rishi et al.: On the Opportunities and Risks of Foundation Models. In: *CoRR* abs/2108.07258 (2021).

Bowman, Samuel R., Gabor Angeli, et al.: A large annotated corpus for learning natural language inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. Ed. by Lluís Màrquez et al. The Association for Computational Linguistics, 2015, pp. 632–642.

Bowman, Samuel R., Jon Gauthier, et al.: A Fast Unified Model for Parsing and Sentence Understanding. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

Brown, Tom B. et al.: Language Models are Few-Shot Learners. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

Cann, Ronnie: Formal semantics: An introduction. Cambridge: Cambridge University Press, 1993.

Carlsson, Fredrik et al.: Semantic Re-tuning with Contrastive Tension. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Cer, Daniel et al.: Universal Sentence Encoder for English. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations,*

Brussels, Belgium, October 31 - November 4, 2018. Ed. by Eduardo Blanco and Wei Lu. Association for Computational Linguistics, 2018, pp. 169–174.

Chen, Daoyuan et al.: AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 2463–2469.

Chen, Liu et al.: Tree-LSTM Guided Attention Pooling of DCNN for Semantic Sentence Modeling. In: *5G for Future Wireless Networks - First International Conference, 5GWN 2017, Beijing, China, April 21-23, 2017, Proceedings*. 2017, pp. 52–59.

Chen, Qian et al.: Enhanced LSTM for Natural Language Inference. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 2017, pp. 1657–1668.

Chen, Ting et al.: A Simple Framework for Contrastive Learning of Visual Representations. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607.

Cho, Kyunghyun et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1724–1734.

Choi, Jihun, Kang Min Yoo, and Sang-goo Lee: Learning to Compose Task-Specific Tree Structures. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 5094–5101.

Chomsky, Noam: Three models for the description of language. In: *IRE Trans. Inf. Theory* 2.3 (1956), pp. 113–124.

Chomsky, Noam: Syntactic structures. Mouton, 1957.

Chu, Yoeng-Jin: On the shortest arborescence of a directed graph. In: *Scientia Sinica* 14 (1965), pp. 1396–1400.

Clark, Kevin et al.: What Does BERT Look at? An Analysis of BERTś Attention. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019*. Ed. by Tal Linzen et al. Association for Computational Linguistics, 2019, pp. 276–286.

Cohan, Arman et al.: SPECTER: Document-level Representation Learning using Citation-informed Transformers. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky et al. Association for Computational Linguistics, 2020, pp. 2270–2282.

Collobert, Ronan and Jason Weston: A unified architecture for natural language processing: deep neural networks with multitask learning. In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Ed. by William W. Cohen, Andrew McCallum, and Sam T. Roweis. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 160–167.

Conneau, Alexis and Douwe Kiela: SentEval: An Evaluation Toolkit for Universal Sentence Representations. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. Ed. by Nicoletta Calzolari et al. European Language Resources Association (ELRA), 2018.

Conneau, Alexis, Douwe Kiela, et al.: Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Association for Computational Linguistics, 2017, pp. 670–680.

Conneau, Alexis, Germán Kruszewski, et al.: What you can cram into a single \$&!# vector: Probing sentence embeddings for linguistic properties. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 2126–2136.

Conneau, Alexis and Guillaume Lample: Cross-lingual Language Model Pretraining. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 7057–7067.

Conneau, Alexis, Ruty Rinott, et al.: XNLI: Evaluating Cross-lingual Sentence Representations. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 2475–2485.

Coster, William and David Kauchak: Simple English Wikipedia: A New Text Simplification Task. In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*. The Association for Computer Linguistics, 2011, pp. 665–669.

Craswell, Nick et al.: MS MARCO: Benchmarking Ranking Models in the Large-Data Regime. In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz et al. ACM, 2021, pp. 1566–1576.

Csordás, Róbert, Kazuki Irie, and Jürgen Schmidhuber: The Devil is in the Detail: Simple Tricks Improve Systematic Generalization of Transformers. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Ed. by Marie-Francine Moens et al. Association for Computational Linguistics, 2021, pp. 619–634.

Cummins, Robert: Systematicity. In: *Journal of Philosophy* 93.12 (1996), pp. 591–614.

Dagan, Ido et al.: Recognizing textual entailment: Rational, evaluation and approaches - Erratum. In: *Nat. Lang. Eng.* 16.1 (2010), p. 105.

Dahl, Deborah A. et al.: Expanding the Scope of the ATIS Task: The ATIS-3 Corpus. In: *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994*. Morgan Kaufmann, 1994.

Dasgupta, Ishita et al.: Evaluating Compositionality in Sentence Embeddings. In: *Proceedings of the 40th Annual Meeting of the Cognitive Science Society, CogSci 2018, Madison, WI, USA, July 25-28, 2018*. Ed. by Chuck Kalish et al. cognitivesciencesociety.org, 2018.

Davidson, Donald: Truth and meaning. In: *Synthese* 17.1 (1967), pp. 304–323.

Dehghani, Mostafa et al.: Universal Transformers. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Devlin, Jacob et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186.

Dowty, David: Compositionality as an empirical problem. In: *In Chris Barker and Pauline Jacobson (eds.) Direct Compositionality*. 2007, pp. 23–101.

Dozat, Timothy and Christopher D. Manning: Deep Biaffine Attention for Neural Dependency Parsing. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Dua, Dheeru et al.: DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 2368–2378.

Duchi, John C., Elad Hazan, and Yoram Singer: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In: *J. Mach. Learn. Res.* 12 (2011), pp. 2121–2159.

Dunn, Matthew et al.: SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. In: *CoRR* abs/1704.05179 (2017).

Eddine, Moussa Kamal, Antoine J.-P. Tixier, and Michalis Vazirgiannis: BARThez: a Skilled Pretrained French Sequence-to-Sequence Model. 2020.

Edmonds, Jack et al.: Optimum branchings. In: *Journal of Research of the national Bureau of Standards B* 71.4 (1967), pp. 233–240.

Eger, Steffen, Andreas Rücklé, and Iryna Gurevych: Pitfalls in the Evaluation of Sentence Embeddings. In: *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*. Ed. by Isabelle Augenstein et al. Association for Computational Linguistics, 2019, pp. 55–60.

Elbayad, Maha et al.: Depth-Adaptive Transformer. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Enard, Mathias: Zone. Actes sud, 2008.

Ettinger, Allyson et al.: Assessing Composition in Sentence Vector Representations. In: *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. Ed. by Emily M. Bender, Leon Derczynski, and Pierre Isabelle. Association for Computational Linguistics, 2018, pp. 1790–1801.

Fader, Anthony, Luke Zettlemoyer, and Oren Etzioni: Open question answering over curated and extracted knowledge bases. In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Ed. by Sofus A. Macskassy et al. ACM, 2014, pp. 1156–1165.

Fan, Angela, Edouard Grave, and Armand Joulin: Reducing Transformer Depth on Demand with Structured Dropout. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Fan, Angela, Yacine Jernite, et al.: ELI5: Long Form Question Answering. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 3558–3567.

Fan, Angela, Mike Lewis, and Yann N. Dauphin: Hierarchical Neural Story Generation. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 889–898.

Feng, Fangxiaoyu et al.: Language-agnostic BERT Sentence Embedding. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Association for Computational Linguistics, 2022, pp. 878–891.

Filippova, Katja and Yasemin Altun: Overcoming the Lack of Parallel Data in Sentence Compression. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2013, pp. 1481–1491.

Finegan-Dollak, Catherine et al.: Improving Text-to-SQL Evaluation Methodology. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 351–360.

Frege, Gottlob: Über sinn und bedeutung. In: *Zeitschrift für Philosophie und philosophische Kritik* 100 (1892). translated as 'On Sense and Reference' by M. Black in Geach and Black (eds. and trans.), pp. 25–50.

Frege, Gottlob: Letter to Jourdain. Gottfried Gabriel et al. ranslated in Philosophical and Mathematical Correspondence, Hans Kaal (trans.) Chicago University Press, 1914, pp. 78–80.

Furrer, Daniel et al.: Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. In: *CoRR* abs/2007.08970 (2020).

Geva, Mor, Ankit Gupta, and Jonathan Berant: Injecting Numerical Reasoning Skills into Language Models. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky et al. Association for Computational Linguistics, 2020, pp. 946–958.

Glorot, Xavier and Yoshua Bengio: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. Ed. by Yee Whye Teh and D. Mike Titterington. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256.

Graves, Alex: Adaptive Computation Time for Recurrent Neural Networks. In: *CoRR* abs/1603.08983 (2016).

Gulordava, Kristina et al.: Colorless Green Recurrent Networks Dream Hierarchically. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, 2018, pp. 1195–1205.

Guo, Mandy et al.: Effective Parallel Corpus Mining using Bilingual Sentence Embeddings. In: *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*. Ed. by Ondrej Bojar et al. Association for Computational Linguistics, 2018, pp. 165–176.

Hamilton, William L.: Graph Representation Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.

Han, Lushan et al.: UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In: *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*. 2013, pp. 44–52.

Hauser, Marc D, Noam Chomsky, and W Tecumseh Fitch: The faculty of language: what is it, who has it, and how did it evolve? In: *science* 298.5598 (2002), pp. 1569–1579.

Heffernan, Kevin, Onur Çelebi, and Holger Schwenk: Bitext Mining Using Distilled Sentence Representations for Low-Resource Languages. In: *CoRR* abs/2205.12654 (2022).

Henderson, Matthew, Pawel Budzianowski, et al.: A Repository of Conversational Datasets. In: *CoRR* abs/1904.06472 (2019).

Henderson, Matthew, Paweł Budzianowski, et al.: A Repository of Conversational Datasets. In: *Proceedings of the Workshop on NLP for Conversational AI*. Data available at github.com/PolyAI-LDN/conversational-datasets. 2019.

Hidey, Christopher and Kathy McKeown: Identifying Causal Relations Using Parallel Wikipedia Articles. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

Hill, Felix, Kyunghyun Cho, and Anna Korhonen: Learning Distributed Representations of Sentences from Unlabelled Data. In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. The Association for Computational Linguistics, 2016, pp. 1367–1377.

Hochreiter, Sepp and Jürgen Schmidhuber: Long Short-Term Memory. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

Hodosh, Micah, Peter Young, and Julia Hockenmaier: Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. In: *J. Artif. Intell. Res.* 47 (2013), pp. 853–899.

Honnibal, Matthew and Mark Johnson: An Improved Non-monotonic Transition System for Dependency Parsing. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. Ed. by Lluís Màrquez et al. The Association for Computational Linguistics, 2015, pp. 1373–1378.

Hoover, Benjamin, Hendrik Strobelt, and Sebastian Gehrmann: exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformers Models. In: *CoRR* abs/1910.05276 (2019).

Hou, Lu et al.: DynaBERT: Dynamic BERT with Adaptive Width and Depth. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

Hugo, Victor: Les Misérables. A. Lacroix, Verboeckhoven & Cie., 1862.

Hupkes, Dieuwke et al.: Compositionality Decomposed: How do Neural Networks Generalise? In: *J. Artif. Intell. Res.* 67 (2020), pp. 757–795.

Jawahar, Ganesh, Benoît Sagot, and Djamé Seddah: What Does BERT Learn about the Structure of Language? In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 2019, pp. 3651–3657.

Jurafsky, Dan and James H. Martin: Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 3rd Edition. 2022. Forthcoming.

Kamath, Aishwarya and Rajarshi Das: A Survey on Semantic Parsing. In: *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*. 2019.

Kamp, Hans and Uwe Reyle: From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. 42. Springer Science & Business Media, 1993.

Kaya, Yigitcan, Sanghyun Hong, and Tudor Dumitras: Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3301–3310.

Keysers, Daniel et al.: Measuring Compositional Generalization: A Comprehensive Method on Realistic Data. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Khashabi, Daniel et al.: GooAQ: Open Question Answering with Diverse Answer Types. In: *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*. Ed. by Marie-Francine Moens et al. Association for Computational Linguistics, 2021, pp. 421–433.

Kim, Najoung and Tal Linzen: COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 9087–9105.

Kiros, Ryan et al.: Skip-Thought Vectors. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes et al. 2015, pp. 3294–3302.

Kitaev, Nikita and Dan Klein: Constituency Parsing with a Self-Attentive Encoder. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 2676–2686.

Klein, Dan and Christopher D. Manning: Accurate Unlexicalized Parsing. In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan: Association for Computational Linguistics, July 2003, pp. 423–430.

Koehn, Philipp et al.: Moses: Open Source Toolkit for Statistical Machine Translation. In: *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. Ed. by John A. Carroll, Antal van den Bosch, and Annie Zaenen. The Association for Computational Linguistics, 2007.

Kong, Fang and Guodong Zhou: Combining Dependency and Constituent-based Syntactic Information for Anaphoricity Determination in Coreference Resolution. In: *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation, PACLIC 25, Singapore,*

December 16-18, 2011. Ed. by Helena Hong Gao and Minghui Dong. Digital Enhancement of Cognitive Development, Waseda University, 2011, pp. 410–419.

Koupaee, Mahnaz and William Yang Wang: WikiHow: A Large Scale Text Summarization Dataset. In: *CoRR* abs/1810.09305 (2018).

Kryscinski, Wojciech et al.: Neural Text Summarization: A Critical Evaluation. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 540–551.

Kwiatkowski, Tom et al.: Natural Questions: a Benchmark for Question Answering Research. In: *Trans. Assoc. Comput. Linguistics* 7 (2019), pp. 452–466.

Lacoste, Alexandre et al.: Quantifying the Carbon Emissions of Machine Learning. In: *CoRR* abs/1910.09700 (2019).

Lai, Alice and Julia Hockenmaier: Illinois-LH: A Denotational and Distributional Approach to Semantics. In: *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*. Ed. by Preslav Nakov and Torsten Zesch. The Association for Computer Linguistics, 2014, pp. 329–334.

Lake, Brenden M et al.: Building machines that learn and think like people. In: *Behavioral and brain sciences* 40 (2017).

Lake, Brenden M. and Marco Baroni: Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2879–2888.

Lample, Guillaume and François Charton: Deep Learning For Symbolic Mathematics. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Lan, Zhenzhong et al.: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. 2020.

Le, Hang et al.: FlauBERT : des modèles de langue contextualisés pré-entraînés pour le français (FlauBERT : Unsupervised Language Model Pre-training for French). In: *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 2 : Traitement Automatique des Langues Naturelle, Nancy, France, June 8-19, 2020*. Ed. by Christophe Benzitoun et al. ATALA, 2020, pp. 268–278.

Le, Hang et al.: FlauBERT: Unsupervised Language Model Pre-training for French. In: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille,*

France, May 11-16, 2020. Ed. by Nicoletta Calzolari et al. European Language Resources Association, 2020, pp. 2479–2490.

Le, Quoc V. and Tomás Mikolov: Distributed Representations of Sentences and Documents. In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. Vol. 32. JMLR Workshop and Conference Proceedings. JMLR.org, 2014, pp. 1188–1196.

Lewis, Patrick S. H. et al.: PAQ: 65 Million Probably-Asked Questions and What You Can Do With Them. In: *CoRR* abs/2102.07033 (2021).

Li, Jiwei et al.: Visualizing and Understanding Neural Models in NLP. In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. The Association for Computational Linguistics, 2016, pp. 681–691.

Lin, Chin-Yew: Rouge: A package for automatic evaluation of summaries. In: *Text summarization branches out*. 2004, pp. 74–81.

Lin, Tsung-Yi et al.: Microsoft COCO: Common Objects in Context. In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. Ed. by David J. Fleet et al. Vol. 8693. Lecture Notes in Computer Science. Springer, 2014, pp. 740–755.

Linzen, Tal and Marco Baroni: Syntactic Structure from Deep Learning. In: *CoRR* abs/2004.10827 (2020).

Linzen, Tal, Emmanuel Dupoux, and Yoav Goldberg: Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. In: *Trans. Assoc. Comput. Linguistics* 4 (2016), pp. 521–535.

Linzen, Tal and Brian Leonard: Distinct patterns of syntactic agreement errors in recurrent networks and humans. In: *Proceedings of the 40th Annual Meeting of the Cognitive Science Society, CogSci 2018, Madison, WI, USA, July 25-28, 2018*. Ed. by Chuck Kalish et al. cognitivesciencesociety.org, 2018.

Liu, Weijie et al.: FastBERT: a Self-distilling BERT with Adaptive Inference Time. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky et al. Association for Computational Linguistics, 2020, pp. 6035–6044.

Liu, Yang, Matt Gardner, and Mirella Lapata: Structured Alignment Networks for Matching Sentences. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 1554–1564.

Liu, Yinhan et al.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: *CoRR* abs/1907.11692 (2019).

Lo, Kyle et al.: S2ORC: The Semantic Scholar Open Research Corpus. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky et al. Association for Computational Linguistics, 2020, pp. 4969–4983.

Locke, John: An essay concerning human understanding. Kay & Troutman, 1847.

Logeswaran, Lajanugen and Honglak Lee: An efficient framework for learning sentence representations. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Loshchilov, Ilya and Frank Hutter: Decoupled Weight Decay Regularization. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Loula, João, Marco Baroni, and Brenden M. Lake: Rearranging the Familiar: Testing Compositional Generalization in Recurrent Networks. In: *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*. Ed. by Tal Linzen, Grzegorz Chrupala, and Afra Alishahi. Association for Computational Linguistics, 2018, pp. 108–114.

Maillard, Jean, Stephen Clark, and Dani Yogatama: Jointly learning sentence embeddings and syntax with unsupervised Tree-LSTMs. In: *Nat. Lang. Eng.* 25.4 (2019), pp. 433–449.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz: Building a Large Annotated Corpus of English: The Penn Treebank. In: *Computational Linguistics* 19.2 (1993), pp. 313–330.

Marelli, Marco et al.: A SICK cure for the evaluation of compositional distributional semantic models. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*. Ed. by Nicoletta Calzolari et al. European Language Resources Association (ELRA), 2014, pp. 216–223.

Martin, Louis et al.: CamemBERT: a Tasty French Language Model. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky et al. Association for Computational Linguistics, 2020, pp. 7203–7219.

Marvin, Rebecca and Tal Linzen: Targeted Syntactic Evaluation of Language Models. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 1192–1202.

McCoy, Tom, Ellie Pavlick, and Tal Linzen: Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 2019, pp. 3428–3448.

McInnes, Leland and John Healy: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. In: *CoRR* abs/1802.03426 (2018).

Micikevicius, Paulius et al.: Mixed Precision Training. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Mikolov, Tomás, Kai Chen, et al.: Efficient Estimation of Word Representations in Vector Space. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2013.

Mikolov, Tomás, Ilya Sutskever, et al.: Distributed Representations of Words and Phrases and their Compositionality. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges et al. 2013, pp. 3111–3119.

Mitchell, Melanie: Why AI is Harder Than We Think. In: *CoRR* abs/2104.12871 (2021).

Montague, Richard: Universal Grammar. In: *Theoria* 36.3 (1970), pp. 373–398.

Montague, Richard: The Proper Treatment of Quantification in Ordinary English. In: *Approaches to Natural Language*. Ed. by K. J. J. Hintikka, J. Moravcsic, and P. Suppes. 1973, pp. 221–242.

Nadeem, Moin, Anna Bethke, and Siva Reddy: StereoSet: Measuring stereotypical bias in pretrained language models. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*. Ed. by Chengqing Zong et al. Association for Computational Linguistics, 2021, pp. 5356–5371.

Naik, Aakanksha et al.: Exploring Numeracy in Word Embeddings. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 3374–3380.

Nangia, Nikita et al.: CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 1953–1967.

Narayanan, Deepak et al.: Efficient large-scale language model training on GPU clusters using megatron-LM. In: *SC '21: The International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021*. Ed. by Bronis R. de Supinski, Mary W. Hall, and Todd Gamblin. ACM, 2021, 58:1–58:15.

Newman, Benjamin et al.: Refining Targeted Syntactic Evaluation of Language Models. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11,*

2021. Ed. by Kristina Toutanova et al. Association for Computational Linguistics, 2021, pp. 3710–3723.

Ney, Hermann, Ute Essen, and Reinhard Kneser: On structuring probabilistic dependences in stochastic language modelling. In: *Comput. Speech Lang.* 8.1 (1994), pp. 1–38.

Nie, Allen, Erin Bennett, and Noah Goodman: DisSent: Learning Sentence Representations from Explicit Discourse Relations. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 2019, pp. 4497–4510.

Nie, Yixin, Yicheng Wang, and Mohit Bansal: Analyzing Compositionality-Sensitivity of NLI Models. In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 6867–6874.

Oepen, Stephan et al.: SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. In: *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. Ed. by Daniel M. Cer et al. The Association for Computer Linguistics, 2015, pp. 915–926.

Ontañón, Santiago et al.: Making Transformers Solve Compositional Tasks. In: *CoRR* abs/2108.04378 (2021).

Oord, Aäron van den, Yazhe Li, and Oriol Vinyals: Representation Learning with Contrastive Predictive Coding. In: *CoRR* abs/1807.03748 (2018).

Ortiz Suárez, Pedro Javier, Benoît Sagot, and Laurent Romary: Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. en. In: ed. by Piotr Bański et al. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019. Mannheim: Leibniz-Institut für Deutsche Sprache, 2019, pp. 9–16.

Partee, Barbara et al.: Compositionality. In: *Varieties of formal semantics* 3 (1984), pp. 281–311.

Pasupat, Panupong and Percy Liang: Compositional Semantic Parsing on Semi-Structured Tables. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 1470–1480.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning: Glove: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. ACL, 2014, pp. 1532–1543.

Peters, Matthew E. et al.: Dissecting Contextual Word Embeddings: Architecture and Representation. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*

Processing, Brussels, Belgium, October 31 - November 4, 2018. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 1499–1509.

Proust, Marcel: Sodome et Gomorrhe. Gallimard, 1921.

Qu, Yingqi et al.: RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Ed. by Kristina Toutanova et al. Association for Computational Linguistics, 2021, pp. 5835–5847.

Radford, Alec, Karthik Narasimhan, et al.: Improving Language Understanding by Generative Pre-Training. OpenAI Blog : Improving Language Understanding with Unsupervised Learning. 2019.

Radford, Alec, Jeff Wu, et al.: Language Models are Unsupervised Multitask Learners. OpenAI Blog : Better Language Models and Their Implications. 2019.

Rajpurkar, Pranav, Robin Jia, and Percy Liang: Know What You Don't Know: Unanswerable Questions for SQuAD. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 784–789.

Ravishankar, Vinit et al.: Attention Can Reflect Syntactic Structure (If You Let It). In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Ed. by Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty. Association for Computational Linguistics, 2021, pp. 3031–3045.

Reimers, Nils and Iryna Gurevych: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 3980–3990.

Reimers, Nils and Iryna Gurevych: Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 4512–4525.

Robinson, Joshua David et al.: Contrastive Learning with Hard Negative Samples. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Russin, Jake et al.: Compositional generalization in a deep seq2seq model by separating syntax and semantics. In: *CoRR* abs/1904.09708 (2019).

Sanh, Victor et al.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In: *CoRR* abs/1910.01108 (2019).

Saunshi, Nikunj et al.: A Theoretical Analysis of Contrastive Unsupervised Representation Learning. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5628–5637.

Saxton, David et al.: Analysing Mathematical Reasoning Abilities of Neural Models. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Schuster, Sebastian and Christopher D. Manning: Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. Ed. by Nicoletta Calzolari et al. European Language Resources Association (ELRA), 2016.

Sennrich, Rico, Barry Haddow, and Alexandra Birch: Neural Machine Translation of Rare Words with Subword Units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

Sheng, Emily et al.: The Woman Worked as a Babysitter: On Biases in Language Generation. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 3405–3410.

Shi, Haoyue et al.: On Tree-Based Neural Sentence Modeling. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 4631–4641.

Shoeybi, Mohammad et al.: Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. In: *CoRR* abs/1909.08053 (2019).

Simoulin, Antoine and Benoit Crabbé: Contrasting distinct structured views to learn sentence embeddings. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*. Online: Association for Computational Linguistics, Apr. 2021, pp. 71–79.

Simoulin, Antoine and Benoit Crabbé: How Many Layers and Why? An Analysis of the Model Depth in Transformers. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*. Online: Association for Computational Linguistics, Aug. 2021, pp. 221–228.

Simoulin, Antoine and Benoit Crabbé: Un modèle Transformer Génératif Pré-entrainé pour le _____ français (Generative Pre-trained Transformer in _____ (French). In: *Actes de la 28e*

Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale. Lille, France: ATALA, June 2021, pp. 246–255.

Socher, Richard, Jeffrey Pennington, et al.: Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2011, pp. 151–161.

Socher, Richard, Alex Perelygin, et al.: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2013, pp. 1631–1642.

Sohn, Kihyuk: Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al. 2016, pp. 1849–1857.

Song, Kaitao et al.: MPNet: Masked and Permuted Pre-training for Language Understanding. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

Stolcke, Andreas: SRILM - an extensible language modeling toolkit. In: *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*. Ed. by John H. L. Hansen and Bryan L. Pellom. ISCA, 2002.

Strobelt, Hendrik et al.: LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. In: *IEEE Trans. Vis. Comput. Graph.* 24.1 (2018), pp. 667–676.

Subramanian, Sandeep et al.: Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Sundararaman, Dhanasekar et al.: Methods for Numeracy-Preserving Word Embeddings. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 4742–4753.

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning: Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 1556–1566.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick: BERT Rediscovers the Classical NLP Pipeline. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*

2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 4593–4601.

Thawani, Avijit et al.: Representing Numbers in NLP: a Survey and a Vision. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Ed. by Kristina Toutanova et al. Association for Computational Linguistics, 2021, pp. 644–656.

Tian, Yonglong, Dilip Krishnan, and Phillip Isola: Contrastive Multiview Coding. In: Lecture Notes in Computer Science 12356 (2020). Ed. by Andrea Vedaldi et al., pp. 776–794.

Tiedemann, Jörg: Parallel Data, Tools and Interfaces in OPUS. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*. Ed. by Nicoletta Calzolari et al. European Language Resources Association (ELRA), 2012, pp. 2214–2218.

Tschannen, Michael et al.: On Mutual Information Maximization for Representation Learning. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Tschannen, Michael et al.: On Mutual Information Maximization for Representation Learning. In: (2020).

Vaswani, Ashish et al.: Attention is All you Need. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008.

Voita, Elena et al.: Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 5797–5808.

Wallace, Eric et al.: Do NLP Models Know Numbers? Probing Numeracy in Embeddings. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 5306–5314.

Wang, Alex, Yada Pruksachatkun, et al.: SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 3261–3275.

Wang, Alex, Amanpreet Singh, et al.: GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 2019.

Wang, Bin and C.-C. Jay Kuo: SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models. In: *IEEE ACM Trans. Audio Speech Lang. Process.* 28 (2020), pp. 2146–2157.

Wang, Wenhui et al.: MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

Wang, Yau-Shian, Hung-yi Lee, and Yun-Nung Chen: Tree Transformer: Integrating Tree Structures into Self-Attention. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 1061–1070.

Wieting, John and Kevin Gimpel: ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, 2018, pp. 451–462.

Wieting, John and Douwe Kiela: No Training Required: Exploring Random Encoders for Sentence Classification. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Williams, Adina, Andrew Drozdov, and Samuel R. Bowman: Do latent tree learning models identify meaningful structure in sentences? In: *Trans. Assoc. Comput. Linguistics* 6 (2018), pp. 253–267.

Williams, Adina, Nikita Nangia, and Samuel R. Bowman: A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, 2018, pp. 1112–1122.

Wittgenstein, Ludwig: Philosophical investigations. Macmillan, 1953.

Wolf, Thomas et al.: Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*. Ed. by Qun Liu and David Schlangen. Association for Computational Linguistics, 2020, pp. 38–45.

Wu, Zhirong et al.: Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. In: *CoRR* abs/1805.01978 (2018).

Xin, Ji et al.: DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online,*

July 5-10, 2020. Ed. by Dan Jurafsky et al. Association for Computational Linguistics, 2020, pp. 2246–2251.

Yang, Yinfei, Gustavo Hernández Ábrego, et al.: Improving Multilingual Sentence Embedding using Bi-directional Dual Encoder with Additive Margin Softmax. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 5370–5378.

Yang, Yinfei, Daniel Cer, et al.: Multilingual Universal Sentence Encoder for Semantic Retrieval. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*. Ed. by Asli Celikyilmaz and Tsung-Hsien Wen. Association for Computational Linguistics, 2020, pp. 87–94.

Yogatama, Dani et al.: Learning to Compose Words into Sentences with Reinforcement Learning. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

You, Yang et al.: Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Young, Peter et al.: From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. In: *Trans. Assoc. Comput. Linguistics* 2 (2014), pp. 67–78.

Zelle, John M. and Raymond J. Mooney: Learning to Parse Database Queries Using Inductive Logic Programming. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*. Ed. by William J. Clancey and Daniel S. Weld. AAAI Press / The MIT Press, 1996, pp. 1050–1055.

Zhang, Xiang, Junbo Jake Zhao, and Yann LeCun: Character-level Convolutional Networks for Text Classification. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes et al. 2015, pp. 649–657.

Zhang, Yuan, Jason Baldridge, and Luheng He: PAWS: Paraphrase Adversaries from Word Scrambling. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. 2019, pp. 1298–1308.

Zhang, Zhuosheng et al.: Semantics-Aware BERT for Language Understanding. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 9628–9635.

Zhou, Jie et al.: Graph neural networks: A review of methods and applications. In: *AI Open* 1 (2020), pp. 57–81.

Zhou, Wangchunshu et al.: BERT Loses Patience: Fast and Robust Inference with Early Exit. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

Zhou, Yao, Cong Liu, and Yan Pan: Modelling Sentence Pairs with Tree-structured Attentive Encoder. In: *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. Ed. by Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad. ACL, 2016, pp. 2912–2922.

Zhu, Yukun et al.: Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 19–27.