Learning to identify and encode entities with deep learning models

Ionut-Teodor Sorodoc

TESI DOCTORAL UPF / Year 2021

THESIS SUPERVISOR
Gemma Boleda
Department of Translation and Language Sciences

Acknowledgements

My first thanks are to my supervisor Gemma Boleda for her support and advice during the last four years. Thank you, Gemma, for the wise guidance both through bad and good moments, successes and failures. I would also like to thank my committee members: Raquel Fernandez, Sebastian Riedel and Anna Rogers, who took the time to read and assess my work. Also, I was very lucky to have Laura Aina as my academic twin, who shared the same road as me during these years, and she was always there to listen to me and support me.

During my PhD, I was incredibly honoured to work with a lot of amazing researchers. I am particularly grateful to Kristina Gulordava and Marco Baroni for all the insightful discussions and for sharing their knowledge and research passion with me. A bit more broadly, I would like to thank all the people that I collaborated with in the past couple of years: Sandro Pezzelle, Raffaella Bernardi, Carina Silberer, Matthijs Westera, German Kruszewski, Tomas Mikolov, Alba Herrera, Carles Ventura and Xavier Giro.

Besides the people mentioned above, there are several others who, in different ways, played an important role during the years of my PhD, and helped me reach this achievement. First of all, Marco Del Tredici and Sandro Pezzelle were always there for brainstorming and for giving feedback throughout the whole PhD, from designing the first experiments to writing the thesis. Also, I am thankful to all the people that I met at Universitat Pompeu Fabra: Lucas, Giorgia, Xixian, Dominika, Alice, Roberto, Eleonora, Elia, Kata, Raquel, Erendira and Thomas for their help in many aspects of the PhD.

The research I present in this thesis was made possible thanks to funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154) and from the Catalan government (SGR 2017 1575), and thanks to the computer resources at CTE-POWER from the Barcelona Supercomputing Center (RES-IM2019- 3-0006; RES-FI-2018-3-0034).

On a personal note, I want to thank the friends from all over the world that made me feel that I belong in every place I lived in during the past years. Also, I am very grateful to all my old friends from my hometown that always listened to me, welcomed me whenever I went back and kept the notion of home strong in me.

Last but not least, I would like to say *multumesc* to my parents, Petru and Eufrozina for always being there.

Abstract

In this thesis, I tackle the ability of deep neural networks to represent entities, and I assess the extent to which this feature impacts tasks involving entities. I consider two standard architectures, LSTM and Transformer, both for analysis and as the main components of the developed models. First, I investigate the behaviour of different model components in a controlled setup, and then I probe the referential information encoded in these models when they are trained on language modelling. Using the insights from the analysis experiments, I develop a set of models and I test their performance on the task of character identification. I show that, while the models achieve good results on this task, the entity representations developed by them are not at the same level. Through different analyses conducted on these models, I investigate how the task, the models and the data impact this difference between task performance and entity representations.

Resum

En aquesta tesi, abordo la capacitat de les xarxes neuronals profundes per representar entitats, i avaluo fins a quin punt aquesta característica afecta les tasques que impliquen entitats. Incloc dues arquitectures estàndard, LSTM i Transformer, tant per a l'anàlisi com per al desenvolupament de models computacionals. En primer lloc, investigo el comportament de diferents components dels models en un entorn controlat, i examino la informació referencial codificada en aquests models quan s'entrenen com a models de llenguatge. A continuació, utilitzant els resultats d'aquestes anàlisis, desenvolupo un conjunt de models i poso a prova el seu rendiment en la tasca d'identificació de personatges. Demostro que, tot i que els models aconsegueixen bons resultats en aquesta tasca, les representacions d'entitats que construeixen aquests models no es troben al mateix nivell. A través de diferents anàlisis, investigo com la tasca, els models i les dades afecten aquesta diferència entre el rendiment en la tasca i les representacions d'entitats que emergeixen.

Resumen

En esta tesis, abordo la capacidad de las redes neuronales profundas para representar entidades, y evalúo hasta qué punto esta característica afecta las tareas que involucran entidades. Considero dos arquitecturas estándar, LSTM y Transformer, tanto para el análisis como para el desarrollo de modelos computacionales. Primero, investigo el comportamiento de diferentes componentes de los modelos en un entorno controlado, y a continuación examino qué información referencial está codificada en estos modelos cuando se entrenan como modelos de lenguaje. Usando los resultados de estos análisis, desarrollo un conjunto de modelos y examino su rendimiento en la tarea de identificación de personajes. Muestro que, si bien los modelos logran buenos resultados en esta tarea, las representaciones de entidades desarrolladas por los mismos no están al mismo nivel. A través de diferentes análisis, investigo cómo la tarea, los modelos y los datos impactan esta diferencia entre el rendimiento en la tarea y las representaciones de entidades.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Motivation

An important component of human communication is the ability to use language to refer to entities that are part of our life, either persons or objects. As shown in Figure 1.1, the text from the bubble is uttered by a speaker, and it refers to real-world entities. Modelling the process of using linguistic expressions to refer to the external world is an important factor for computationally processing language. For example, conversational agents became an important component of our life, like virtual assistants (e.g. Alexa) or GPS navigation systems, and these systems require a good grasp of the external world in order to have a coherent dialogue.

We aim to build a performant computational model of reference to entities. We use deep learning methods when we develop the computational models in this dissertation.

These models are very powerful, achieving state of the art results on the majority of computational linguistics tasks (Goldberg, 2017) like: sequence tagging (İrsoy and Cardie, 2014; Ling et al., 2015), parsing (Dyer et al., 2015; Zhou et al., 2015) or machine translation (Cho et al., 2014). A very important task that was revolutionized by deep learning models is neural language modelling (Bengio et al., 2003). As shown in Figure 1.2, in the case of language modelling, the model needs to predict what is the next word in a sequence, given the previous uttered words as input. The deep learning models are able to do this task very well through learning rich lexical representations and capturing the linguistic context of utterances.

While these models develop very competent linguistic representations, they also manifest a few weaknesses that impact our modelling goals. First, they mostly focus on the information encoded in the linguistic input from Figure 1.1 (snippet of the dialogue from the sitcom "Friends"), ignoring all the other compo-

Figure 1.1: Example of a dialogue from the TV show "Friends".

nents that connect the language to the real world [1]. The text is not informative if we don't create the links to the external world where the words are produced by a speaker in a specific situation referring to a set of individuals. For example, the language model from Figure 1.2 understands that the next word is supposed to be a word describing a human, but it doesn't incorporate other types of information, such as speaker or interlocutors, in order to consider "guy" as the most viable option. Furthermore, they struggle with processing specific situations, where we require fast recognition of individual elements, such as entities, events, and relationships, and the ability to reason about them (Bernardi et al., 2015).

Looking at the example from Figure 1.1, in order to understand to which entities the dialogue refers to, we need to be able to capture patterns like: "his" refers usually to an entity introduced earlier by a more direct referential expression or that the word "I" refers to the speaker. On the other hand, we want our model to learn that the name "Ross" refers to a specific entity and that Julie is the partner of Ross, which is more in the realm of having good general entity representations. From another perspective, the model should have a good blend between global semantic knowledge about entities and contextual knowledge (Kamp and Reyle,

[1] there is research using vision (Lin et al., 2014; Antol et al., 2015; Das et al., 2017), but few works focused on entities (Lu et al., 2018)

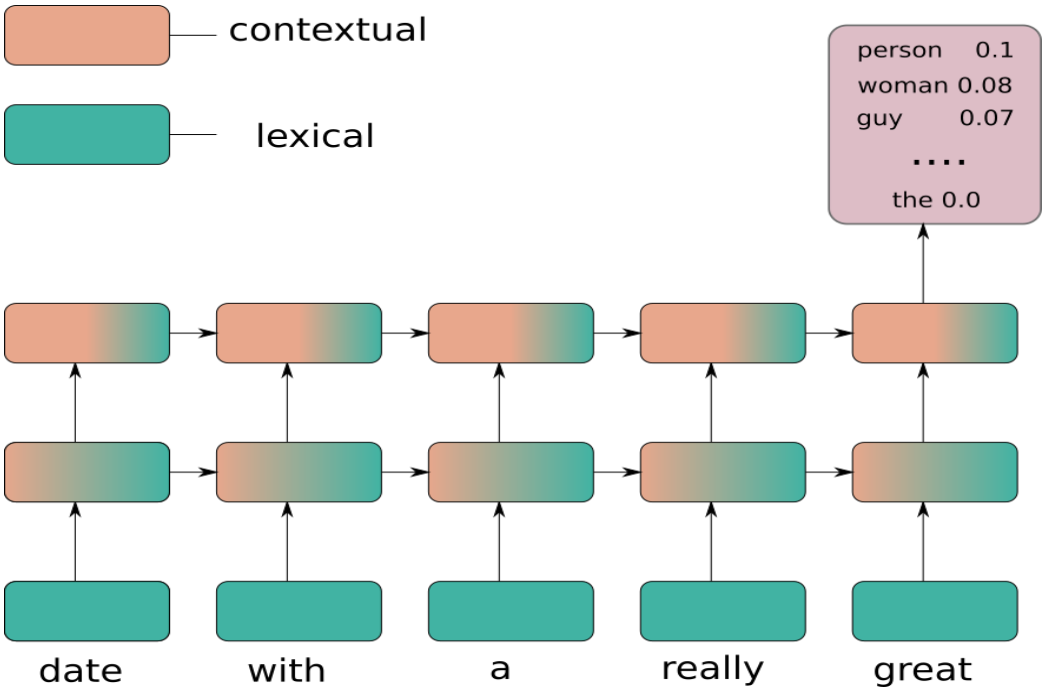

Figure 1.2: Neural network example (neural language model) - "lexical" refers to generic information associated with a word type, "contextual" refers to information from the linguistic context

2013). For example, "guy" is an attribute that is generally associated with half of the characters mentioned in the dialogue, but by analyzing both the linguistic and extra-linguistic context, the model should infer that in this case, the character referred to with this expression is Michael who is the only male character from the scene.

The goal of this dissertation is to define a computational model with the ability to develop this range of features for the entities that it is exposed to. In order to achieve this, we consider that we need a model that captures **linguistic context** (e.g. anaphoric patterns), **extra-linguistic information** (e.g. speaker or interlocutor awareness) and **rich semantic representations of entities** (specific characteristics of people/objects).

Even though the deep learning models are very performant and they are end to end, going from raw input to the desired predictions, they don't have a transparent view of how the information is encoded inside neural networks or which types of patterns are developed, in order to solve a task. This drawback led to the creation of a new type of research, called blackbox analysis which aims to a better understanding of how recent computational models work. Understanding how a system behaves is a prerequisite to improving it, so the thesis focuses first

on analyzing to what extent discrete information and more specifically, referential information is encoded in these models. Following the findings of these experiments, we develop methods to improve these architectures regarding the aspect of reference and entity representation.

1.2 Approach

Our approach for achieving the goals of this dissertation is to combine analysis and computational system modelling to develop a model which handles reference to entities.

The models analysed and developed in this thesis are artificial neural networks (Rosenblatt, 1958). More specifically, they are deep learning models (LeCun et al., 2015) which are built of multiple processing layers to learn representations of data with multiple levels of abstraction. This type of algorithms revolutionized in the last ten years fields like natural language processing (NLP), computer vision (Krizhevsky et al., 2012) and speech processing (Hinton et al., 2012). These architectures are part of the representation learning paradigm, which aims to detect patterns from the data and induce a series of representations as they learn to perform a task. In order to achieve this, they compose a series of successive transformations of the input, which are modified through various optimization algorithms.

1.2.1 Basic architectures

Currently, the field of NLP has two standard deep learning architectures that are used on most of the tasks. These models constitute the main components of all the models analyzed or developed in this thesis.

First, Long Short Term Memory networks (LSTMs (Hochreiter and Schmidhuber, 1997)) are models that were designed to process sequences. Considering that the linguistic input is sequential, this type of architecture is a natural fit for Computational Linguistics. LSTMs belong to the family of Recurrent Neural Networks (RNNs: Elman (1990)). Given a sequence, the RNN model computes a hidden state at each time step as a function of the input vector and the vector representing the previous hidden state. The main problem of RNNs is that it is very facile to forget the information from the initial inputs, especially in long sequences.

This problem was overcome in LSTMs by a memory cell which should prevent the information from the beginning of the sequence to disappear. Furthermore, the network functions based on a gating mechanism that controls which information

and how much information is added to the memory cell and to the hidden state at each time step.

LSTMs were the standard architecture in NLP until the development of the attention mechanism (Bahdanau et al., 2014) and its associated architecture: Transformer (Vaswani et al., 2017).

Transformers are currently the most successful architecture for NLP. While the memory cell of the LSTM should prevent the information from the first items of the sequence to disappear, LSTM still has problems with conserving information in long sequences because all the past information is compressed in the memory cell with no further access to the input representation. Transformer overcomes this problem by getting rid of the recursion; the text is processed as a whole and not sequentially anymore. The main component of the Transformer is the self-attention mechanism, which is an attention mechanism connecting different positions of a sequence in order to compute a representation of the sequence. For each input, the model creates a contextual representation that takes into account the other inputs in the sequence relative to the current one following a learned criterion. Because the model is not recursive anymore, we can also use parallel computation, which facilitates scaling this architecture to multiple layers and attention mechanisms.

1.2.2 Enhancements

Even though LSTMs and Transformers are the backbone of deep learning in NLP, their basic structure is not enough in many cases. These models are very powerful when they have a multitude of layers, which requires a big amount of data to learn from when they are trained from scratch. In some cases, it is either very hard to get more data for a task or the cost of annotation is too high. These limitations stimulated the development of model enhancements that can overcome the data problem. The main two techniques that we will use in this thesis for overcoming the data are: transfer learning (Ruder, 2019) and additional modules enhancements.

Transfer learning The limitations brought by the amount of annotated data in many NLP tasks was overcome in recent years through transfer learning (Ruder, 2019). This approach is widely used in model development in NLP and it was adopted for many tasks such as question answering, natural language inference, sentiment analysis or textual entailment (Devlin et al., 2018).

Transfer learning is the technique of first training the neural network on a different task (e.g. language modelling) which has access to more data for supervision and then fine-tune the model on the task of interest. While we might not have enough annotated data to learn complex linguistic patterns, we can use a

task that doesn't require annotation like neural language modelling (Bengio et al., 2003) in order to train big neural networks on a big amount of data. Language modelling is the task of predicting the probability of a word in the text, considering its context as input (see Figure 1.2. This procedure will result in a model that encodes a big range of complex linguistic phenomena in its hidden representation as a result of learning the task of language modelling. While training the original model requires a big amount of data and computational resources, their adaptation and fine-tuning to a different linguistic task is not as costly as the training of the model on language modelling. This change in model development resulted in a set of pre-trained language models that are universally used as linguistic encoding components in most of the NLP tasks. This approach was first introduced with Word2Vec (Mikolov et al., 2013a), and it culminated in recent years with contextualized architectures that are either based on LSTM like ELMO (Peters et al., 2018b) or on Transformer like BERT (Devlin et al., 2018). We use Word2Vec to represent the lexical information in Chapter 4, and we incorporate BERT in the proposed model for better coverage of the contextual patterns in Chapter 5.

Additional modules A different way to improve deep learning models is to enhance them with some additional modules that will bias the type of patterns that are learned by the network.

These additional modules can encode patterns that are outside of the scope of the basic architectures and are more tailored towards task-specific patterns. One of the first systems that used this type of module, in the form of an external structure, is the Memory Network (Sukhbaatar et al., 2015) which introduced attention mechanisms over large external memories. Sukhbaatar et al. (2015) show that the addition of a specialized structure improves the performance on multiple tasks like question answering or reading comprehension. Closer to the goals of this thesis, the recurrent entity network (EntNet - Henaff et al. (2016)) is an entity-centric computational model with an external memory. The model contains a fixed number of memory cells, each formed by a vector key and a vector value. The cells can be seen as slots for an entity where the key can behave like an identifier and the values as a storage for all the known information about the associated entity. In Chapter 4, we experiment with two different dedicated structures for encoding entities: a static variant inspired by the memory network and a dynamic entity library inspired by EntNet. The purpose of these structures is to facilitate biasing the semantic knowledge and contextual knowledge (in the case of the dynamic library) towards entities and to better capture the extra-linguistic information.

1.2.3 Analysis

When we want to capture complex phenomena in our representations, such as referential information, we require a big number of parameters and transformations (layers). This results in big models that behave like a blackbox (Linzen et al., 2019; Alishahi et al., 2020). It is very hard to understand how the information is encoded because the model goes from raw data to a class output without a transparent view of intermediate reasoning.

More important, language follows Zipf's law (Zipf, 1949) with the most frequent words counting for a big portion of the data, and consequently specific linguistic patterns counting for a big percentage of an NLP task. For example, in the character identification task, 44% of the referential expressions are first-person pronouns and 28% are second-person pronouns. An accuracy of 72% on character identification doesn't tell us if the model learned multiple patterns to some extent or if it focused on capturing the frequent phenomena. Furthermore, it was shown that deep learning models amplify this effect, developing heuristics that focus only on the frequent patterns from the training data (Agrawal et al., 2016; Sanchez et al., 2018; McCoy et al., 2019). All these factors lead to the fact that the general results on NLP tasks don't depict a complete picture of the phenomena captured by the model. These drawbacks of deep learning models lead to more granular reporting of results, and it caused the development of multiple analysis techniques (Belinkov and Glass, 2019; Rogers et al., 2020).

This thesis will use multiple analysis techniques in order to give a more thorough view of the referential information encoded in computational systems. The main two methods that we adapt for our area of interest are called: probing tasks (Conneau et al., 2018a) and challenge sets (Lehmann et al., 1996). We also use techniques like representation similarity analysis (RSA Kriegeskorte et al. (2008)) and t-Distributed Stochastic Neighbor Embedding (t-SNE Van der Maaten and Hinton (2008)) for a better understanding of model behaviour.

Diagnostic classifiers In order to probe the information encoded in the hidden representations of neural networks, the most common type of analysis method is known as "diagnostic classifiers" (Veldhoen et al., 2016) or "probing tasks" (Conneau et al., 2018a).

The main method for disentangling the information in a neural network is the following: we have a model trained on a task like language modelling. Then we use a task with annotation for the linguistic phenomenon that we want to analyze. Using the previously trained model, we generate representations for the input of the second task then a classifier is used to predict the answer for the second task. The performance of the second classifier reflects the performance of the generated representations, so the reasoning is that it will also reflect the performance of

the original model on the studied phenomenon. This type of analysis method is known as "diagnostic classifiers" (Veldhoen et al., 2016) or "probing tasks" (Conneau et al., 2018a). Some examples of linguistic phenomena which were studied using this technique are: part of speech (Shi et al., 2016), number agreement (Giulianelli et al., 2018a), parsing or semantic roles (Tenney et al., 2019a). We use this approach to get a better understanding of which referential information is captured by pre-trained language models (Chapter 3).

Challenge sets From a different perspective, most of the benchmark datasets in NLP are generated from text corpora, reflecting a natural frequency distribution of language phenomena. Even though they are useful in practice for evaluating model performance in the average case, these datasets might fail to capture a wide range of phenomena. This weakness of benchmarks was treated from the early days of NLP with controlled datasets, known as challenge sets (Lehmann et al., 1996). This type of datasets saw a resurgence in recent years, in areas like natural language inference (Poliak et al., 2018) and machine translation (Sennrich, 2017).

In order to develop a model that efficiently captures entity information, we first want to have a better understanding of how the information related to entities is currently encoded in neural networks, so we conduct a series of analysis studies on referential information in neural networks. We simplify the referential task and decompose it into a set of challenge sets (Lehmann et al., 1996) focusing on aspects like feature extraction, contextualization, or order tracking. This allows us to analyze standard NLP architectures (LSTM and Transformer) and zoom into their different components in a controlled environment (Chapter 2). Furthermore, we develop an adjacent challenge set for the task of character identification in Chapter 4, which we use for probing the degree of semantic knowledge about entities in the developed models.

1.3 Goals of the thesis

This thesis has the following two goals:

1. To analyze the entity encoding capabilities of neural networks. We want to understand how current computational models, and more specifically neural networks, capture features relevant to tasks involving entities.

We approach this research question from two different angles:

- what is the capacity of the current standard neural architectures to encode patterns relevant to our model development, such as feature extraction, contextualization or order tracking?

- which patterns related with entities emerge when training a neural network on a more general task such as language modelling?

2. To develop computational models with the capacity to encode both linguistic and extra-linguistic referential information. Our goal is to improve NLP models on entity-centric tasks, more specifically, character identification. We use the insights from the analysis conducted in the first part in order to develop a complete model that can combine the contextual patterns with the more global semantic entity representations.

We aim to reach the second goal in two steps:

- develop different specialized structures that can be attached to standard neural networks in order to address the problems discovered in the analysis that we conducted beforehand.
- incorporate the newly developed structure into a pre-trained model in order to have an architecture that combines the benefits of both component features: the linguistic knowledge encoded in pre-trained models with the entity-focused design of specialized structures.

1.4 Structure

The rest of the dissertation is organized as follows:

Chapter 2 presents a set of controlled tasks that are used for analysing how different phenomena relevant to entity encoding such as feature mapping, contextualization or order are processed by neural networks and which components play a crucial role in capturing these patterns.

The content of this chapter is based on the following publication:

Ionut-Teodor Sorodoc, Gemma Boleda and Marco Baroni. 2021. Controlled tasks for model analysis: Retrieving discrete information from sequences. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, to appear.

Chapter 3 presents an analysis methodology to probe the entity-related information encoded in neural network language models.

The content of this chapter is based on the following publication:

Ionut-Teodor Sorodoc, Kristina Gulordava and Gemma Boleda. 2020. Probing for referential information in language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Chapter 4 develops models for solving a character identification task and analyzes which entity patterns are learned and used for solving the task.

The content of this chapter is based on the following publications:

Laura Aina, Carina Silberer, Ionut-Teodor Sorodoc, Matthijs Westera and Gemma Boleda. 2018. AMORE-UPF at SemEval-2018 Task 4: BiLSTM with Entity Library. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. **Winning system** in the SemEval-2018 Task 4 competition.

Laura Aina, Carina Silberer, Ionut-Teodor Sorodoc, Matthijs Westera and Gemma Boleda. 2019. What do Entity-Centric Models Learn? Insights from Entity Linking in Multi-Party Dialogue. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Chapter 5 combines the structures designed for entities from Chapter 4 with a contextualized language model component, similar to the ones analyzed in Chapter 3.

At the time of writing, the content of this chapter has not been published.

Chapter 6 presents a summary of the dissertation and the conclusions.

Chapter 2

ANALYSIS OF FEATURE ENCODING USING CONTROLLED SEQUENTIAL TASKS

2.1 Introduction

When we want to model entity-related information, an important mechanism is the detection and extraction of information that refers to a given entity. In the following experiments, we emulate this process through challenge sets built by a multitude of controlled tasks. Each task is a simplified version of a linguistic pattern relevant for representing entity information. For example, we can refer to an entity through a proper noun (e.g. we can refer to the entity **Ross** using the expressions: "Ross" or "Ross Geller"), a set of attributes (e.g. **Ross** can be referred as "the male paleontologist") or a relationship with somebody else (the expression "Monica's brother" refers to the entity **Ross**). In order for a computational model to be able to handle these patterns, it needs to be able to do feature detection and extraction, contextualization or order tracking.

There has been a continuous increase in performance in computational linguistics in recent years. This development correlates with larger and more complex models, which are trained on ever bigger datasets. These new characteristics of modelling made it harder to understand which model components are relevant and how they work. This flaw has motivated the NLP community to develop new methods for model analysis.

Given the difficulties of analyzing large models, some of this work (Hupkes et al., 2018a; Lake and Baroni, 2018; Hewitt and Liang, 2019; Chrupała and Al-

ishahi, 2019; White and Cotterell, 2021) has focused on controlled tasks that emulate specific aspects of language. We propose a new set of such controlled tasks to explore a crucial aspect of natural language processing for dealing with entities: the need to retrieve discrete information from sequences. [1]

The controlled tasks are built using binary vectors as input in order to exclude the possible inferences of linguistic patterns that are not the focus of our analysis. In particular, we design the tasks such that the models need to emulate four abilities that are crucial for natural language in general and entities related tasks in particular: incremental processing, indirect mappings, contextualization, and order tracking. The insights from these experiments will impact our further model developments used for character identification.

We study model behaviour on the proposed tasks with simple instantiations of Transformers (Vaswani et al., 2017) and LSTMs (Hochreiter and Schmidhuber, 1997), and show that, for most of the tasks, these models show significant difficulties. Transformers are state of the art in deep learning for NLP, and LSTMs are a classical architecture that was designed for sequence processing. In the analysis, we aim at understanding the role of the different components of the models, focusing on self-attention, decoder attention and positional encoding for Transformers and decoder attention for LSTMs.

2.2 Related work

The current study belongs in the newly developed area of interpretability and explainability of computational linguistics, which seeks to understand how models capture natural language phenomena (Alishahi et al., 2019; Belinkov and Glass, 2019; Rogers et al., 2021).

Generally, deep learning models are trained and evaluated on text following the natural distribution of language. While it is a good strategy to mimic a natural setup, this setup omits or downplays some phenomena. The solution for this drawback was the development of challenge sets (Lehmann et al., 1996), which were defined as test suites that followed principles like: control over data, progressivity, systematicity, the inclusion of negative data and exhaustivity.

These datasets have been used in NLP for a long time (Lehmann et al., 1996) and they saw a resurgence in popularity in recent years. This technique was recently used to study the degree to which different phenomena are captured in standard NLP models, such as coreference in MT systems (Müller et al., 2018), number agreement in language models (Gulordava et al., 2018b), lexical inference in NLI systems (Glockner et al., 2018) or compositionality in seq2seq mod-

[1] The associated dataset and code can be downloaded at `https://github.com/sorodoc/DiscreteSeq`

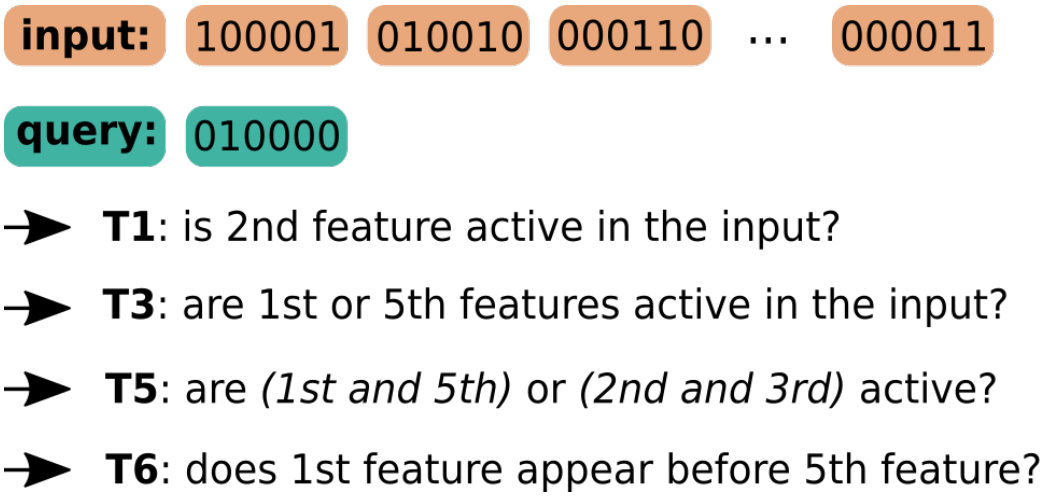

Figure 2.1: Schematic description of our tasks. Different tasks require different interpretations of the query. The answer is positive for all these instances.

els (Lake and Baroni, 2018). Even though this technique is generally applied as a controlled evaluation setup, we use the principles behind it to create complete controlled tasks where we can investigate the capabilities of different architectures in a range of different linguistic phenomena.

Some other work has used controlled tasks to analyze neural networks regarding specific aspects, such as reasoning skills (Weston et al., 2015), compositionality (Lake and Baroni, 2018; Hupkes et al., 2020), PoS tagging (Hewitt and Liang, 2019), inductive biases (White and Cotterell, 2021) or hierarchical structure (Hupkes et al., 2018a; Chrupała and Alishahi, 2019).

We follow this latter methodology, and design tasks that highlight one of the core abilities that a model needs to possess in order to process natural language, namely detecting one or more features in a sequence of tokens, possibly in a context- and/or order-sensitive way.

2.3 Description of the tasks

2.3.1 Task description

In all tasks, the goal is to perform *feature detection*, that is, to provide a binary response about whether a feature (or feature combination) is present in the input sequence. The tasks are schematically illustrated in Figure 2.1. The model is always presented with an *input* consisting in a sequence of *tokens*. Each token is a 36-dimensional binary vector with a variable number of dimensions (*fea-*

tures) activated (set to 1). The model is also given a *query*, which is a single 36-dimensional binary vector with one feature activated, except for Task 2 below, where it has two activated features.

The tasks are meant to be *incremental*, in the sense that, when processing the input sequence, the model does not know what information it will need to retrieve from it. Some uses of Transformers in the NLP literature instead give access to all the information from the start, such as in BERT (Devlin et al., 2019) and its variants. Arguably, incrementality is necessary for most instances of language understanding "in the wild": For some applications, like Machine Translation for documents, it is realistic to give access to all the input at once, but as we move towards real-time applications such as virtual assistants, models will need to act incrementally.

T1: one-feature detection. T1 asks whether the active feature of the query is present in some token in the sequence. A linguistic example, relevant for syntactic processing, would be: did the plural feature occur in a span of tokens?

In the example in Figure 2.1, the second input token has feature 2, which is the one the query asks about, so the answer should be positive.

T2: two-feature detection. T2 asks whether two features occur in the sequence, be it in the same or in different tokens. This ability is required, for example, to check agreement between two syntactic units, or to answer a conjoint question.

In T1 and T2, query features and input features coincide: if the query asks about feature 2, then feature 2 needs to be active in some token for the answer to be positive. In the rest of the tasks, the relationship between query and input features is instead indirect. For instance, as illustrated in Figure 2.1 for T3, a query with feature 2 may ask about features 1 and 5 in the input. Models need to learn this implicit mapping when they are trained. Translation is an example (among many) of an indirect linguistic task, where words in the source language map to different target-language words.

T3: set member detection. In T3, models need to detect whether at least one out of two features is present in the sequence, that is, the task checks for feature disjunction (see Figure 2.1). This is akin to set member detection because we ask for a positive answer when at least one of the two features (set members) is present.

For example, many question-answering setups require retrieving an *instance* (*Fido, Snoopy, ...*) when prompted with the name of the *class* (*dog*).

T4: contextualized 2-feature detection. T4 asks about a conjunction of two features, like T2 (but with the indirect mapping). A linguistic example could involve a question requiring more than one piece of information to answer (*Who played the 1982 World Cup Final? Italy and Germany*). In addition, we design the mapping such that each feature is part of 2 queries; hence, it is associated only with two other features. This dataset structure should encourage **contextualization**, because a given feature is only ever relevant in the context of one of the other two features it is associated with.

Contextualization is crucial for natural language, where the same feature (e.g., a word) requires different responses depending on other features occurring in the context. A natural hypothesis is that the Transformer is naturally good for contextualization (this should be the strength of self-attention), but as we will see our results are mixed.

T5: contextualized set member detection. Like T3, the task concerns disjunction with indirect mapping, but it consists in checking whether at least one out of two *pairs* of features is present in the input (e.g., $(1 \wedge 5) \vee (2 \wedge 7)$; see Figure 2.1). Again, in each pair, one feature is only a hit in the context of the other, so this task also requires contextualization.

A linguistic example would be a coreference task where *toy* might refer to *rubber lion* or *plastic truck*, but not to *lion* or *truck* in other contexts.

T6: ordered feature detection. T6 adds order to T4: it asks about a conjunction of two features with the indirect mapping, where the features must be in a specific order. This has many linguistic counterparts, e.g., parsing words in the correct order to determine their syntactic relation.

T7: contextualized ordered set member detection. Finally, T7 adds order to T5: It asks models to check whether at least one out of two pairs of features is present in the input, but now the order of context matters (e.g., *(1 before 5)* $\vee$ *(2 before 7)*). An example would be semantic role identification, where *cow* is typically an *agent* if it precedes *eats*, but not if it follows it.

Entity correspondence. We use these tasks for insight regarding model development for character identification. They are relevant because they emulate proper noun detection of an entity with the direct tasks, the third-person pronoun or common noun detection with the contextualized tasks or relation detection with the ordered tasks.

	Nr. of distinct queries	Nr. of queries containing each feature
T1	36	1
T2	630	36
T3, T4, T6	36	2
T5, T7	36	4

Table 2.1: Description of the query space.

2.3.2 Dataset construction

All datasets contain 100k training, 10k validation, and 10k test examples. For each datapoint, we sample uniformly at random whether the answer is positive or negative. For each task, we create datasets with sequence lengths 5, 10, 15, 20, 25, and 30. Also, for each datapoint, we exclude between 1 and 6 randomly chosen attributes. We apply this restriction because otherwise, for longer sequences, nearly all positive datapoints would contain all features, whereas negative datapoints must, by construction, miss at least one feature. This would allow models to develop a degenerate guessing strategy ("datapoint is positive if it contains all features").

T1. For each datapoint, we choose a random feature as a query. If the answer is positive, the feature will appear in the sequence. All other features are randomly set to active with $p = 0.2$. During training, the queried feature can appear multiple times (following the 0.2 probability), but it appears only once at test time for two main reasons: to facilitate analysis and to avoid having the number of appearances of the queried feature in the sequence as an informative variable.

T2. The 2 queried features are randomly chosen. This is the only task where we have 2 query activations, instead of 1. As is highlighted in Table 2.1, the query space is much larger for this task, in comparison with all the other tasks (630 possible queries, vs. 36 possible queries for the others). For positive datapoints, we randomly choose whether the features are in the same token. For negative datapoints, we randomly choose whether one of the features appears in the sequence.

For T3-7, we use predefined mappings between query and input features (these mappings will not be accessible to the models, but will need to be learned as part of carrying out the task). The input feature pairs in these tasks consist of a feature between 1 and 18 and one between 19 and 36. Also, we aim for an equal

representation of features in the query space, so each feature appears twice in T3, T4 and T6 and four times in T5 and T7 in the input-query mapping (see Table 2.1).

T3. First, we choose a random feature to activate in the query. We then check which are the 2 features associated with the chosen query feature in the predefined mapping. Then, for positive datapoints, we choose randomly whether one or 2 of these features appear in the sequence. If 2 features appear, we randomly choose whether they appear in the same token. For negative datapoints, we ensure that none of these 2 features appear in the input sequence.

T4. The pipeline is similar to T2, the difference being that we randomly choose 1 query and we use its associated features.

T5. We choose one of the 2 pairs of features that are associated with one query. We then apply the same pipeline as in T2 while ensuring that at least 1 feature from the other pair doesn't appear in the sequence.

T6. For positive cases, we randomly choose 2 positions in the sequence and put the 2 tokens on these positions in the correct order. For negative cases, we randomly choose whether the datapoint contains both features in the wrong order or contains at most one of the features.

T7. We choose one of the 2 pairs of features that are associated with one query. We then apply the same procedure as in T6, while ensuring that at least 1 feature from the other pair doesn't appear in the sequence.

2.4 Experiments

2.4.1 Model implementation

Transformer The Transformer architecture consists of a sequence encoder and a decoder adapted to our tasks. We choose the most basic architecture for interpretability: a single-head, single-layer architecture. To enable incremental processing, the encoder self-attention only looks into the past, as in standard "causal" architectures such as the one of Dai et al. (2019a).

The general structure of the model is presented in Figure 2.2. The model was implemented using the framework Pytorch (Paszke et al., 2019).

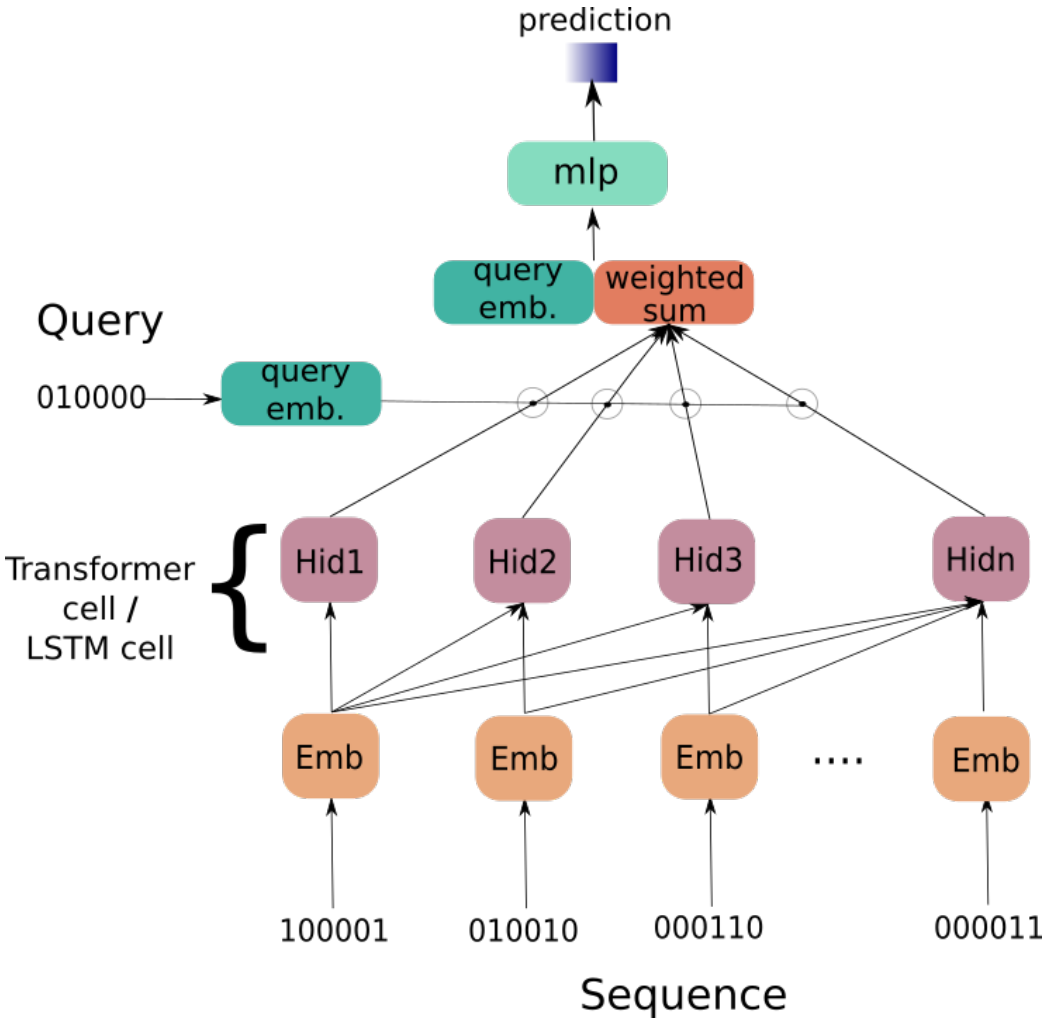

Figure 2.2: Diagram of the complete model.

Each datapoint has **n** 36-dimension binary vectors as input **inp** and a 36-dimension binary vector as query **q**. All these vectors are embedded to 100 dimensions using the matrices W_{in} and W_q. Then we apply a sinusoidal positional encoding on the input embeddings and the ReLU function on top of the query embedding:

$$emb_{in_i} = PosEnc(in_i * W_{in}) \tag{2.1}$$

$$emb_q = ReLU(q * W_q) \tag{2.2}$$

with PosEnc defined for each input position *pos* and for each token dimension *i* similarly to the method used in the main Transformer architecture (Vaswani et al.,

2017):

$$\begin{aligned} PosEnc(pos, 2i) &= sin(pos/10000^{2i/d_{model}}) \\ PosEnc(pos, 2i+1) &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.3}$$

Then, each input embedding goes through the TransformerEncoder cell which has 1 attention head and 1 layer:

$$h_i = TransfEnc(emb_{in_i}) \tag{2.4}$$

as defined in Vaswani et al. (2017), TransfEnc is a scaled dot product attention where d is the dimensionality of the input vectors. The dot product is scaled in order to not have regions with very slow gradients.

$$TransfEnc(X) = softmax(\frac{X * X^T}{\sqrt{d}}) * X \tag{2.5}$$

In order to decode relevant information based on the query we use a dot product attention. α_i represents the attention value that we put on token embedding h_i relative to the query embedding. We then calculate the vector c which is the sum of the token embeddings weighted by α.

$$\alpha_i = \frac{\exp(h_i * emb_q)}{\sum_{k=1}^{n} \exp(h_k * emb_q)} \tag{2.6}$$

$$c = \sum_{i=1}^{n} \alpha_i h_i \tag{2.7}$$

We then use a multi-layer perceptron to generate the answer. Firstly, we apply a dimensionality reduction using W_{o1} from 200 to 100 dimensions with ReLU as nonlinearity on top of it. Then the hidden state hid_o is mapped from 100 to 1 dimension. Using the Sigmoid function, we get our result as a number in the range of [0,1]. At test time, if $o \geq 0.5$, then we consider the answer to be positive.

$$hid_o = ReLU((c||emb_q) * W_{o1}) \tag{2.8}$$

$$o = Sigmoid(hid_o * W_{o2}) \tag{2.9}$$

The model is optimized with a binary cross-entropy loss. At test time, a result larger than 0.5 is considered a positive answer, as is standard.

We ablate the full Transformer architecture by removing self-attention and skipping positional encoding when embedding the input. The variants solely based on decoder attention are akin to Memory Networks (Sukhbaatar et al., 2015).

LSTM LSTM is a sequential neural network. At each time step, it reads an input and it decides how much and which information of the input should be kept on the internal memory and what should be ignored/forgotten. It was a staple architecture for computational linguistics in the years preceding Transformers due to its performance, but also due to its similarities with the human approach of processing language.

In order to have a fair comparison with the Transformer model, we do minimal changes to the architecture, only substituting the transformer self-attention block with the LSTM cell described below.

An LSTM cell is formed by multiple gates which decide the flow of the information. At a given time **t**, the input forget i_t decides how much information from the current step should be used further and the forget gate f_t decides how much information from the previous hidden state is conserved. Then, we update the cell state c_t by multiplying the old state c_{t-1} by f_t, forgetting the things we decided to forget earlier. Then we add the new candidate values g_t, scaled by how much we decided to update each state value i_t. The last step is to create an output vector for the current time step h_t by filtering the relevant information from the cell state c_t using the output gate o_t.

$$i_t = \sigma(W_{ii} * x_t + b_{ii} + W_{hi} * h_{t-1} + b_{hi}) \tag{2.10}$$

$$f_t = \sigma(W_{if} * x_t + b_{if} + W_{hf} * h_{t-1} + b_{hf}) \tag{2.11}$$

$$g_t = tanh(W_{ig} * x_t + b_{ig} + W_{hg} * h_{t-1} + b_{hg}) \tag{2.12}$$

$$o_t = \sigma(W_{io} * x_t + b_{io} + W_{ho} * h_{t-1} + b_{ho}) \tag{2.13}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.14}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.15}$$

We experiment with two variants of LSTM:

- (basic) LSTM: the input goes through the LSTM and the output of the last time step is considered the representation of the input and is concatenated with the query embedding to be sent to the MLP.
- attention LSTM: similarly to the Transformer model, there is an attention mechanism that compares the query embedding to the output from each time step (again using dot product). The input representation that will be

Tasks↓ Models→	T-Pos-Att	T+Pos-Att	T-Pos+Att	T+Pos+Att
T1 1 feature	91.3±2.8	96.1±2.9	96.5±1.7	**98.7**±1.1
T2 2 features	83.4±0.2	**84.7**±0.5	83.2±0.2	**84.9**±1.5
T3 set member	98.7±1.2	**100**	99.4±0.9	**100**
T4 context. 2 features	86.2±0.6	**87.7**±0.3	86.4±0.6	**87.7**±0.2
T5 context. set member	77.4±0.4	77.7±1.2	**78.8**±2.5	77.8±1.1
T6 ordered feature	56.6±1	80.2±0.4	**92.1**±0.3	90.7±1.9
T7 ord. context. set member	57.5±0.4	65.8±0.8	**78.6**±0.6	67.6±0.6

Table 2.2: Results for sequence length 10. Results are averaged over 5 random seeds, with s.d. Best results bold-faced. Models are coded as follows: **+/-Pos** marks absence (-) vs. presence (+) of positional encoding, **+/-Att** marks absence (-) vs. presence (+) of self-attention. The full Transformer is model T+Pos+Att.

concatenated to the query representation is then the weighted sum of the outputs.

We opt for these two variants because the first variant has the capacity of keeping in its hidden state the presence of different features and the position where they were active, but it has problems with long sequences. On the other hand, the second variant has the ability to switch between a recursive architecture to a more independent architecture by closing the forget gate.

2.4.2 Learning procedure

All the experiments are optimized with Adam and a learning rate of 0.0001. We apply 0.2 dropout, gradient clipping at 0.5. We run 100 epochs with batch size 10 and save the model at the epoch with the highest validation accuracy. We experimented with different hyperparameters, but we found the ones we just reported to give the most stable results.

2.5 Results

2.5.1 Transformer

Since the Transformer surpasses the LSTM in most cases, and the Transformer patterns are quite similar across sequence lengths, we first take a detailed look at the behaviour of the Transformer on the tasks for sequence length 10. Table 2.2 summarizes the results for this sequence length. Recall that datapoints are always

uniformly distributed between negative and positive answers, such that a random baseline always averages 50% accuracy.

Task 1. All the models succeed at T1 (single feature detection) with an accuracy of over 90%. As could be expected, there is a high correlation between the accuracy and the degree of decoder attention on the correct token: This attention is at an average of 0.82 when the answer is correct, vs. 0.17 when it is wrong.

Surprisingly, while this task doesn't involve contextualization nor order, both self-attention and positional encoding bring a boost in performance, from 91.3% accuracy for the base model (T-Pos-Att) to over 96% for the models T+Pos-Att and T-Pos+Att (which add positional encoding and self-attention, respectively). We conjecture that positional embeddings act as a beneficial noising mechanism, akin to regularization: altering a token's embedding depending on the position helps the model not to overfit.

We also find that self-attention triggers an inverse recency effect on accuracy: performance is better when the target feature is towards the beginning of the sequence. Indeed, there is a highly significant Pearson correlation of -0.51 between position and accuracy for models with self-attention vs. no significant correlation for models without it. This recency effect is probably due to the fact that self-attention can copy a feature through the following hidden states, so an earlier feature will tend to be more prominent in the weighted sum, and thus easier to detect. We find that this copying mechanism improves results for earlier tokens without significantly harming performance in later tokens.

Thus, the first take-home message from our experiments is that the presence of a component in a Transformer architecture does not imply that the model will learn to use it as expected in a task, even if it puts it to a use that improves performance. In T1, self-attention preserves information, and positional encoding possibly adds noise, and both have a serendipitous positive effect. This underscores the need for model analysis.

Task 2. When moving to two-feature detection (T2), there is a predictable drop in accuracy. The models have problems when the queried features are in different tokens (false negatives), with a 25% accuracy drop, and when only one target feature occurs in the sequence (false positives), with a 40% drop. The main reason why features in different tokens are missed is that decoder attention fails to operate distributively, i.e., to focus on two different input tokens at once: On average, the difference between the decoder attention weight of the most attended token and the second most attended token is of 0.6-0.7 across the models, showing that decoder attention indeed focuses on a single token.

Given that the feature combinations are random, it is not clear how self-attention could help, and indeed using self-attention does not improve results.

Task 3. The T3 results show that all model variants can easily learn to detect a class instance, or feature disjunction, even when using an indirect mapping. Decoder attention suffices, as the models learn to associate a query with both features of its class and trigger high attention values when either of them is present in a token (average attention on target token between 0.6-0.7 for all variants). Here, model behaviour is as expected.

Contextualization. Self-attention should help with contextualized feature detection (T4 and T5), by highlighting a specific feature only when it is preceded by one of the other features it is associated with. However, models settle for a degenerate strategy instead, and thus we detect no competitive advantage for models with self-attention (compare models T-Pos-Att and T-Pos+Att in Table 2.2). In T4, they answer 'yes' if one specific feature is present in the sequence (always the same feature for each query): Average attention for the more attended feature in each query is around 0.9, while the average attention on the other relevant feature is around 0.05. The maximum accuracy for this strategy is 87.5%, which is about what the best models reach in T4.

Thus, the need to contextualize by itself is not enough for models to profit from self-attention; instead, as we will see next, the need to track order does trigger a productive use of self-attention.

Ordered feature detection. In T6 and T7, using self-attention brings accuracy from near chance to 92.1% and 78.6%, respectively (compare models T-Pos-Att and T-Pos+Att in Table 2.2). We find strong evidence that the models with self-attention use it to record the presence of the first feature in the hidden state of the token containing the second, as we predicted it should do already for tasks requiring contextualization: In both T6 and T7, model T-Pos+Att puts most of the decoder attention on the second feature-carrying token in the sequence, as we show next.

We report in Table 2.3 the difference between the average decoder attention on the second vs. first feature-carrying token for T6, T7, and T5, which is the unordered version of T7. In models that use self-attention as predicted, this difference will be large and positive. We see in the first column of the table that for T5, where order does not matter, the difference is very low, while it is much larger for T6 and T7. This confirms that there has been a change in strategy, with self-attention being productively used in the order-sensitive tasks.

	T-Pos+Att	T+Pos+Att
T5	0.03±0.02	0.01±0.01
T6	0.79±0.005	-0.14±0.55
T7	0.46±0.03	0.16±0.18

Table 2.3: Average decoder attention difference (and s.d.) between second and first feature-carrying tokens for the two models with self-attention in the contextualized tasks (T6 and T7), together with T5 for comparison.

Positional encoding is an alternative mechanism to track order, and for T6 and T7 it also improves results substantially with respect to the base models (compare models T-Pos-Att and T+Pos-Att), though much less than self-attention.

However, using both mechanisms actually harms performance (see results for T+Pos+Att). Self-attention and positional encoding seem to get in the way of each other, making it harder, when combined, for the model to converge on a single strategy to track order. We see evidence for this in Table 2.3, where the full Transformer (T+Pos+Att) has a much smaller difference for T7 compared to model T-Pos+Att, and a negative difference for T6 (meaning that it puts more attention on the first feature-carrying token). Moreover, the full Transformers exhibit a large standard deviation in both order-sensitive tasks, which means that different runs converge on different strategies. In contrast, the models with self-attention show a really low standard deviation; they always converge on the expected strategy.

Impact of sequence length. Figure 2.3 presents average accuracies for sequence lengths from 5 to 30 (in steps of 5; error bars represent standard deviations across 5 random seed initializations). The patterns discussed for length 10 are generally confirmed at other sequence lengths. We further notice that longer sequences are beneficial for "easy" tasks, such as T1 and T3 (perhaps they help models avoid trivial guessing strategies). On the other hand, when the complexity of the task is higher (Tasks 5, 6 and 7), longer sequences are detrimental to model performance. Accuracies for tasks for which degenerate solutions were found for sequence length 10 (T2 and T4) do not change across sequence length, indicating that the strategy does not change either. Also, note that for the most difficult tasks (T5-T7), model T-Pos+Att is consistently better in longer sequences. Thus, the "getting in the way" effect observed above holds for the three tasks.

2.5.2 LSTM

In general, the performance of LSTM models on our tasks is markedly worse than that of the Transformer, and they are impacted by sequence length to a much

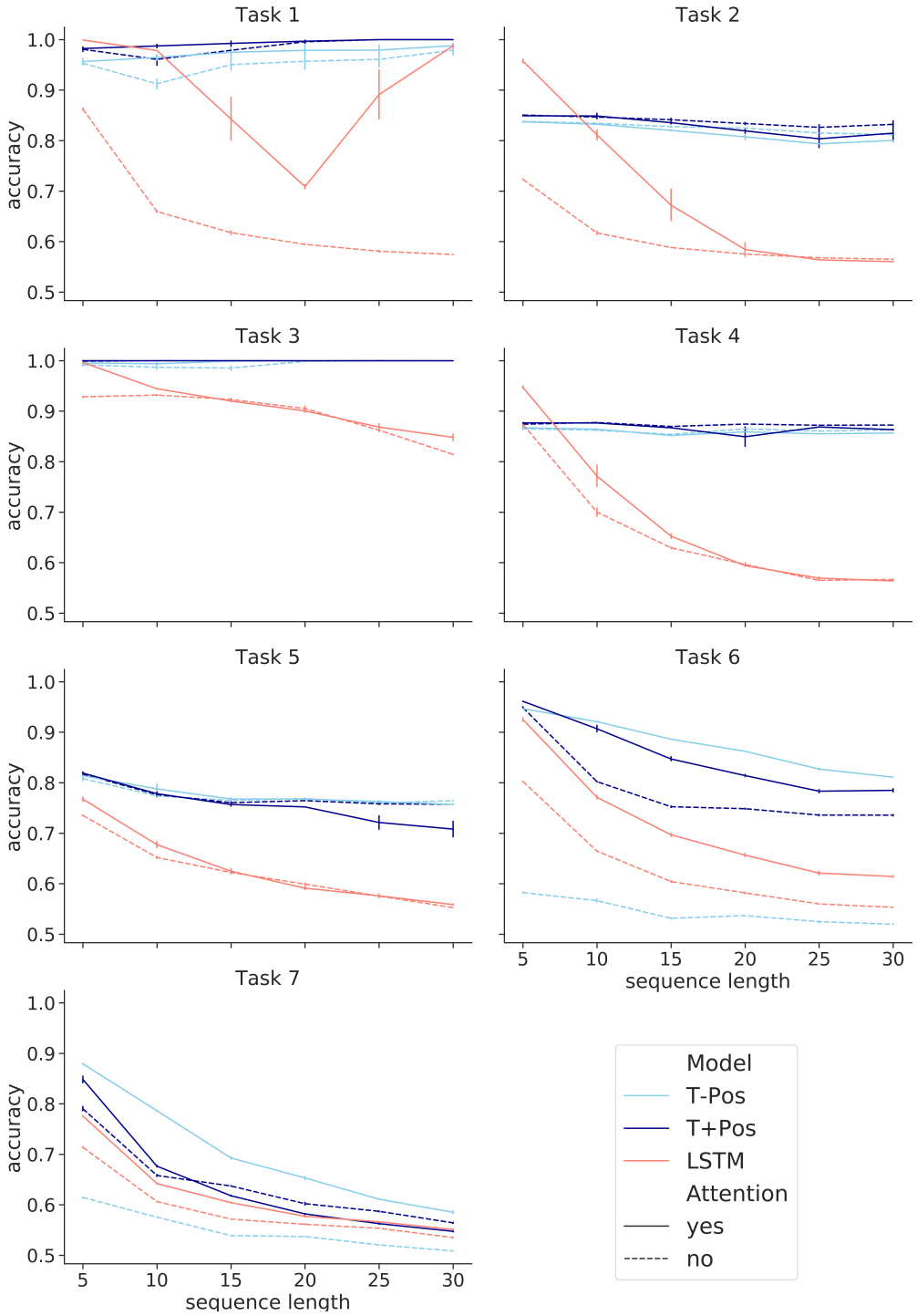

Figure 2.3: Model performance on multiple sequence lengths for all tasks

larger extent (see Figure 2.3). This could be expected given general results on the two architectures (Transformers outperform LSTMs in most computational linguistics tasks, and LSTMs have been shown to have issues with long-distance dependencies). There are two notable exceptions to these general trends.

First, the accuracy of the LSTM with attention in Task 1 shows a characteristic V-shape, dropping at first and recovering for longer sequences (cf. orange solid line in Figure 2.3).[2] The result is that its performance is on par with that of the Transformer models in sequences of lengths 5 and 30.

We find that the LSTM changes its behaviour, from a simple recurrent LSTM to an actual attention-based LSTM. Indeed, for short sequences, it does not use decoder attention to identify the target feature (and still performs optimally): for instance, for sequence length 5, the attention is distributed uniformly, with attention values of around 0.2 on all the input tokens. Instead, for long sequences it does use the attention mechanism: the attention values spike on the position of the input that contains the queried feature (e.g., with an average of 0.85 for sequence length 30). This ability to switch behaviours could theoretically help also for other tasks, but it seems that the complexity of the tasks prevents the model from doing so.

Second, the LSTM with attention surpasses all Transformer variants for short sequences in Tasks 2 and 4. Recall that the Transformer models reached a degenerate solution for these tasks (with a maximum accuracy of 87.5%), in which only one of the two relevant features was attended to; instead, the LSTM solves the task correctly, because the information about the previous tokens flows through the recurrent steps in the token representations.

As for the differences between the two model variants of the LSTM, the model enhanced with the decoder attention is consistently better than the classic, basic model. This suggests that the decoder attention mechanism (also present in all the Transformer models) is beneficial independently of the base architecture.

2.6 Discussion

Our tasks shed light on how the main model components act and interact regarding the retrieval of discrete information from sequences, uncovering behaviours that would be difficult to detect when the architectures are applied to complex NLP tasks.

A take-home message from our experiments is that the presence of a component in an architecture does not imply that the model will learn to use it as expected in a task. In particular, we found that only the need to track ordered

[2]Instead, the basic LSTM model degrades quickly and monotonically with sequence length; see dashed orange line.

information led the architecture to use self-attention in the predicted way, and that decoder attention can be difficult for LSTMs to use correctly when the complexity of the tasks increases, but it is efficient in simple tasks.

On the other hand, the components can assume unexpected functions. Self-attention in transformers can simply serve to blindly propagate information across time, leading to more robust representations of features contained in earlier tokens, that are copied over and over. Also, surprisingly, positional embeddings provide a small but consistent benefit in tasks that do not require order tracking. This suggests that they might have a serendipitous function, possibly adding helpful noise to the representations. Moreover, as both self-attention and positional embeddings can learn to keep track of order, the two mechanisms get in the way of each other, making it harder, when combined, for the Transformer models to converge on a single strategy to track order.

Regarding LSTM, the sequence length is a very big factor. The model has a steep loss in performance when the sequence gets longer. On the simplest task, the model enhanced with decoder attention develops the ability to switch between strategies, but this doesn't generalize to the more complex ones.

To conclude, we hope to have shown that the proposed tasks constitute a useful probing mechanism for the ability of models to detect discrete information in sequences, testing in particular four key abilities: incremental processing (in the sense that the query is not known at input processing time), indirect mappings, context-dependence and order tracking. We have shown that most of the tasks are difficult for models even for short sequences; and the tasks can be easily extended to more dimensions and even longer sequences.

Future work with our controlled setup should go in two different directions: 1) examining how behaviour changes when the probed models are scaled by increasing the number of layers and attention heads, and 2) modifying the models to solve some of the problems that we encounter in the current experiments. We follow the second proposal in our model developments by adding additional modules to the default architectures.

Regarding the impact of these findings in the perspective of modelling reference to entities, we conclude that the decoder attention is a simple and efficient mechanism for identifying and extracting a single feature from a sequence, and it can be placed on top of both Transformer and LSTM. This can prove valuable in the case of simple referential patterns like proper noun entity identification, and also in the case of querying a set of entity representations to identify the specific entity we referred to with our query (which can be the representation of a referential expression). On the other hand, capturing more complex referential phenomena like attributes or relationships to other entities would involve contextualization or order tracking, phenomena that get poor results in our controlled tasks. This conclusion calls for an introduction of some biases towards these referential

phenomena, either through data distribution or through architectural changes. We explore the change of data distribution in Chapter 3 with analysis of pre-trained language models trained on natural data, and we develop architectural changes to the standard models in Chapter 4 where we propose additional modules for better handling of entity information.

Chapter 3

ENCODING REFERENTIAL INFORMATION IN LANGUAGE MODELS

3.1 Introduction

With the previous analysis study, we investigate how standard neural network architectures capture patterns essential to the development of an entity-centric model. Even though these insights are valuable to our model design, a prerequisite for a good entity-centric computational model is a performant component that encodes good linguistic representations, focusing especially on capturing well linguistic context and rich semantic representations of entities.

Neural network-based language models (LMs) provide good general linguistic representation for a wide range of NLP tasks, and they have been shown to encode relevant properties of language without being explicitly trained for them. In particular, recent work suggests that they are able to capture syntactic relations to a large extent (Gulordava et al., 2018b; Kuncoro et al., 2018; Wilcox et al., 2018).

In this chapter, we extend this line of research to analyze whether they are able to capture **referential** aspects of language, focusing on anaphoric relations (pronoun-antecedent relations, as in *she-Yeping Wang* in Figure 3.1).

Previous work, such as Ji et al. (2017), Yang et al. (2017) and Cheng and Erk (2019), showed that augmenting language models with a component that uses an objective based on entity or coreference information improves their performance at language modelling. Intuitively, in the example in Figure 3.1, understanding that the first *she* refers to *Yeping Wang* makes words related to studying or working more likely to follow than other kinds of words. That is, referential information helps language models do their task.

...he$_1$ was elected to be president of the People's Republic of China, and chairman of the$_2$ Central$_2$ Military$_2$ Commission$_2$. Yeping$_3$ Wang$_3$ was born in Shanghai in 1926. She$_3$ studied in Shanghai Foreign Language College, and started working in 1949. For a long time, she$_3$...

Figure 3.1: Example from OntoNotes with a window of 60 tokens (as used in our first probe task). Both occurrences of *she* refer to the same entity as *Yeping Wang*. Note that not all entity mentions are annotated in OntoNotes –only those that enter into coreference relationships in the document.

The cited work includes explicit coreference guidance; however, since referential information is useful for language modelling, we expect language models to learn referential information even without explicit supervision. Here we analyze to what extent this is the case.

We carry out our analysis using probe tasks, or tasks that check whether certain information is encoded in a model (Adi et al., 2016; Linzen et al., 2016; Conneau et al., 2018c; Giulianelli et al., 2018b). The reasoning is as follows: Even if a linguistic property is encoded in the network, it is not necessarily directly accessible through the model output; therefore, we train a probe model to predict a feature of interest, in this case anaphoric coreference, given the model's hidden representations as input.

We focus on two main linguistic levels that are relevant for entity processing: **linguistic context**, with grammatical patterns such as the fact that pronouns agree in number and gender with their antecedents, and **global semantic representation of entities** – the ability to build entity representations that encode entity's defining features and learn to differentiate between two entities based on their representations.

Our hypothesis is that language models will capture grammatical properties, but not semantic entity representations. This hypothesis is based on the observation that the former are formal properties of language that are easier to induce from co-occurrence patterns. The fact that language refers to entities is not obvious from language alone Harnad (1990), and LMs use only textual input. Furthermore, we consider that the lack of explicit components for entities in the LM makes it very difficult for the language model to create specific semantic entity representations.

What we find is that, while it is true that language models are much better at grammar, they do show evidence of learning semantic information to some extent (even though this is overshadowed by factors like proximity). Our explanation for this partially positive result is that, because the same entity underlies all its mentions, the contexts in which the mentions appear are coherent and distinct

from those of mentions of other entities. For instance, in Figure 3.1, the second *she* mention gives additional information about Yeping Wang that is consistent with the information given in the previous sentence.

The contributions brought by the current chapter are the following:

- First, we provide an analysis methodology to probe for referential information encoded in language models, on two kinds of levels: linguistic context, and global semantic representations. This methodology can be applied to any architecture.

- The second contribution is a deeper understanding of the referential capabilities of current language models, and of the differences between Transformers and LSTMs. The Transformer outperforms the LSTM in all the analyses. For the local patterns, the Transformer and the LSTM have the same behavior with a performance difference; instead, they show different behavior with regard to the global semantic information. These last discoveries help us understand the viability of pre-trained language models as a component of an entity-centric model and it also unveils which features are still not present in current models and need to be induced through new model developments.

3.2 Related work

Coreference and anaphora resolution (Mitkov, 2002; Poesio et al., 2016) are among the oldest topics in computational linguistics and have continued to receive a lot of attention in the last decade, as manifested by several shared tasks (Pradhan et al., 2011b, 2012; Poesio et al., 2018). In our analysis we use the OntoNotes dataset Hovy et al. (2006); Pradhan et al. (2012), developed within the coreference resolution community. Our probe tasks are related to coreference resolution; however, our goal is not to train a coreference system but to analyse whether language models extract features relevant for reference without explicit supervision.

A recent line of work has focused on demonstrating that neural networks trained on language modeling, without any linguistic annotation, learn syntactic properties and relations such as agreement or filler-gap dependencies (Linzen et al., 2016; Gulordava et al., 2018b; Kuncoro et al., 2018; Wilcox et al., 2018; Futrell et al., 2018). This is typically done by analysing the predictions of LMs on controlled sets of data. Part of this research uses probe models (also known as diagnostic models) to analyse the information contained in their hidden representations (Adi et al., 2016; Conneau et al., 2018c; Hupkes et al., 2018b; Lakretz et al., 2019; Giulianelli et al., 2018b), as we do here —applying it to referential information.

There is less work on referential information than on syntactic properties such as subject-verb agreement. As for anaphoric reference, Peters et al. (2018a) include a limited test using 904 sentences from OntoNotes. Their results suggest that LMs are able to do unsupervised coreference resolution to a certain extent; our first probe task can be seen as an extended version of their task obtaining more specific insights. Jumelet et al. (2019) analyze the kind of information that LSTM-based LMs use to make decisions in within-sentence anaphora. They find a strong male bias encoded in the network's weights, while the information in the input word embeddings only plays a role in the case of feminine pronouns. We analyze anaphora in longer spans (60 tokens / whole document) and include also a Transformer.

The above work suggests that LMs capture grammatical patterns about anaphora to a large extent. There is much less evidence that LMs can capture a notion of entity, as that which nominal elements refer to, and that they are able to track entities across a discourse. Parvez et al. (2018) show that LSTM-based models have poor results on texts with a high presence of entities; Paperno (2014) that they cannot predict the last word of text fragments that require a context of a whole passage (as opposed to the last sentence only), with data that mostly contain nominal elements. Several models (Henaff et al., 2019; Yang et al., 2017; Ji et al., 2017) were developed as an augmentation of RNN LMs to deal better with entities, with the implicit assumption that standard models do that poorly.

As for Transformer-based architectures, recent research suggests that they give same or better contextualized representations in comparison with LSTM language models, and that they better encapsulate syntactic information (Goldberg, 2019; Wolf, 2019). On the other hand, van Schijndel et al. (2019) show that big Transformer model representations perform on par or even poorer than smaller LSTMs on tasks such as number agreement or coordination, and that, like LSTMs, they have the problem that agreement accuracy decreases as the subject becomes more distant from its verb. Most recent work on analysis of linguistic phenomena in NNs focuses on BERT (Tenney et al., 2019b; Clark et al., 2019; Reif et al., 2019; Broscheit, 2019). In this chapter, we chose to use TransformerXL (Dai et al., 2019b) as our Transformer model, and not BERT, for comparability: We wanted to compare the two most standard architectures for LMs on as equal ground as possible, and the two chosen models, TransformerXL and AWD-LSTM (Merity et al., 2017), share the same training objective and are trained on the same data, with comparable vocabularies.

3.3 Analysis: linguistic context

To shed light into which patterns LMs encode that are useful for coreference, we train a simple anaphora resolution probe model using the hidden layers of LMs as input. By the logic of probe tasks, if the probe model is successful then that means that the relevant information is encoded in the hidden states, and error analysis can provide insight into which kinds of information are available.

3.3.1 Experimental Setup

Data. We train our probe models on data from OntoNotes 5.0 Weischedel et al. (2013). We use the annotated coreference chains, as well as the provided part-of-speech tags (the latter only for analysis purposes).

We take all pronouns that have at least one antecedent in a 60-token context window; the task of the probe model is to identify their antecedent.[1] An example datapoint is provided in Figure 3.1 above (note that a window of 60 tokens allows us to check anaphora beyond the sentence). For simplicity, antecedents are tokens, but typically there is more than one possible token antecedent for a given pronoun: A mention can span several tokens (*Yeping Wang*), and the window can contain several mentions from the same coreference chain (*Yeping Wang* and the first *She* in Figure 3.1); we consider any of the tokens a correct answer. Note that we are not training the model to explicitly identify mentions, their spans or the complete coreference chains, but to identify the tokens that are antecedents of the target pronoun.

To obtain enough data for analysis, especially for low-frequency phenomena, we follow Linzen et al. (2016) in reversing the original partitions of the corpus, using the original test set for training and the original training set for testing. Using little training data has also been shown to lessen the possibility of confounds in the probe model results; in particular, it makes it more difficult for the probe model to exploit regularities in the training data rather than capturing the analyzed model's ability to capture a phenomenon Hewitt and Liang (2019). Voita and Titov (2020) present a theoretical justification from a information-theoretic perspective on this problem. [2]

In addition, we focus on the OntoNotes documents that belong to narrative text sections because the dialogue data does not come with turn segmentation.[3]

[1] We also experimented with windows 20 and 200, obtaining a similar picture.

[2] Results on the original split confirm that the conclusions of the chapter are robust: we see an increase in performance of around 3% overall, as could be expected because we use more data, but the same behavior patterns (on the data that can be compared).

[3] We keep newswire (NW), broadcast news (BN), magazine (MZ), web data (WB), and pivot text (PT), removing broadcast conversation (BC), telephone conversation (TC).

	Tokens	Datapoints
Train	191,830	4,949
Dev	275,201	4,556
Test	2,026,565	45,665

Table 3.1: Dataset statistics for first probe task. We reverse the original train and test partitions (see text).

Resulting data statistics for our task are provided in Table 3.1.

Language models. The base language models we use are AWD-LSTM (Merity et al., 2017) and TransformerXL (Dai et al., 2019b), two models with the most standard architectures for language modeling as of 2021 (LSTM, Transformer). We chose these models for comparison because they are trained on the same dataset (Wiki103; Merity et al., 2016), they have a comparable vocabulary, and they are both very strong language models, with perplexities of 24 for TransformerXL and 33 for AWD-LSTM. TransformerXL is a bit larger than AWD-LSTM, though (151 million parameters compared to 126), which should be kept in mind when assessing results.[4]

Probe model. For each word x_i in the window of size m preceding the target pronoun x_t, we obtain its contextualized representation h_i from the last hidden layer of the language model (Eq. 3.1). The probe model takes this representation as input and is trained to map it onto a vector o_i using a non-linear transformation (Eq. 3.2). The target pronoun representation is transformed in the same way. The dot products between these transformed representations of target and context word vectors give the attention weights ref_i (Eq. 3.3) representing the similarity between two representations. The weights are transformed into probabilities using the softmax function (Eq. 3.4). Like this we obtain a probability distribution p_i over context tokens.

During training, the probe model's objective is to assign higher probabilities (and thus attention weights) to correct antecedents, and lower probabilities to incorrect ones, through the use of the Kullback-Leibler divergence loss (Eq. 3.5). We use the KL loss because we frame the task in terms of a probability distribution over mentions in the context. For the reasons discussed above, there can be

[4]We also trained an in-house LSTM on data that are more similar to those of OntoNotes and a smaller vocabulary. The results for this model (not reported) follow the same patterns as those found for the AWD-LSTM and TransformerXL models, although the performance on this probe task is much higher than that of AWD-LSTM.

$k > 1$ correct predictions out of m tokens in the window. We assume that gold probability distribution is uniform over k correct tokens, that is, each of these tokens has a probability $p_i^* = \frac{1}{k}$ and all other tokens have a probability of 0.[5]

$$h_i = LSTM(x_i) \tag{3.1}$$
$$o_i = ReLU(W * h_i + b) \tag{3.2}$$
$$ref_i = o_i \odot o_t, \forall i \in [t-m, t-1] \tag{3.3}$$
$$p_i = softmax(ref_i), \forall i \in [t-m, t-1] \tag{3.4}$$
$$L = KL(p_i, p_i^*) \tag{3.5}$$

As mentioned above, we fix $m = 60$. We train the probe model for 50 epochs with a learning rate of 1e-5 and ADAM as optimizer. The transformed vectors o_i have a dimensionality of 650 in the case of both models in comparison with h_i which is 400 for the AWD-LSTM and 1024 for TransformerXL.

Baselines. We report two rule-based baselines that give relatively good performance in anaphora resolution: Referring to the previous entity (given by the oracle gold annotation; in Figure 3.1, *she* would refer to the previous *She*), and always pointing to the token in the window that has the same form as the target pronoun (that is, in Figure 3.1, *she* $\rightarrow$ *She* —we ignore capitalization). In addition, to compare the result of the probe model with the input representations, we also report an unsupervised baseline: Referring to the token in the window that has the highest similarity $cos(h_i, h_t)$ to the target pronoun, i.e., relying on the similarity between the non-transformed hidden representations.

3.3.2 Results

Table 3.2 summarizes the results of the pronominal anaphora probe task. The probe model trained on top of the LSTM improves a bit over the strongest baseline, and that of the Transformer does so substantially (75.9 vs. 61.3; the LSTM obtains 64.8). This performance suggests that the LMs use more information than simple heuristics like referring to a token with the same form.

The unsupervised similarity baseline performs worse than the rule-based baselines. This is to be expected: The "raw" similarity between hidden states is based on many more aspects than those related to reference, given that hidden states

[5]Note however that minimizing KL divergence and minimizing cross-entropy gives the same results, because $KLdiv(p||q) = CrossEntropy(p,q) - entropy(p)$, and $entropy(p)$ is constant. Technically, in PyTorch the cross-entropy loss is only implemented for classification task targets, while the more general KL loss is available for predicting probability distributions.

Model	Accuracy	
closest gold entity	56.1	
closest same-form token	61.3	
	unsup.	sup.
LSTM	41.7	64.8
Transformer	48.5	**75.9**

Table 3.2: Probe model results on anaphora resolution.

are responsible for capturing all the contextual features that are relevant for word prediction. This is why a probe model is needed to distill the reference-related information from the hidden layers.

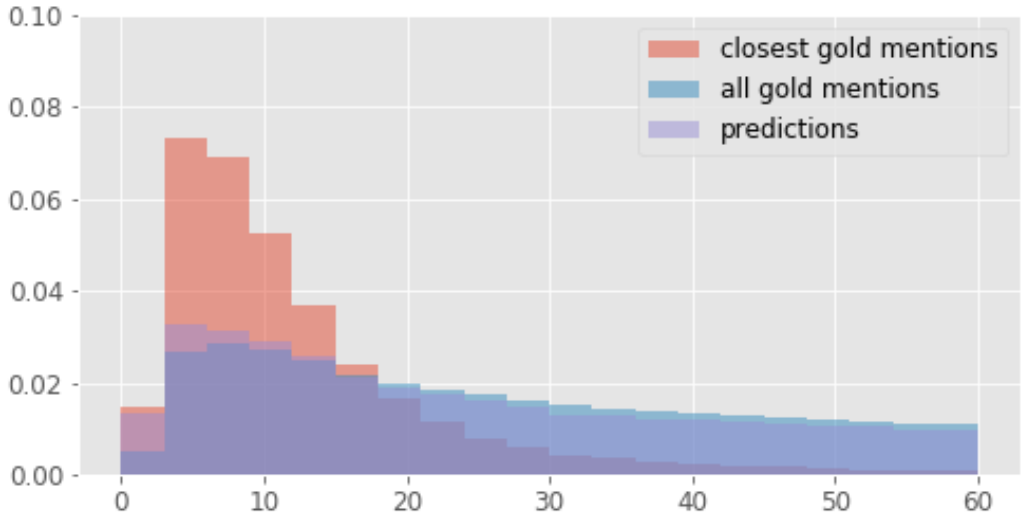

Figure 3.2: The distances between the pronoun and its gold and predicted antecedents for AWD-LSTM.

A single non-linear layer trained on only 5K datapoints improves performance by 23-28 absolute accuracy points (supervised vs. unsupervised results), which suggests that the referential information in the hidden layers is easy to extract. Behaviorally, the unsupervised hidden layers are quite similar to the baselines. First, they are biased towards tokens of the same form: in 27.1% of the cases, the LSTM layer of the pronoun presents the highest similarity to a token with the same form; 29.1% in the case of the Transformer. Second, they prefer close antecedents, although the LSTM presents this recency bias to a much higher degree: in 27.8% of the cases, the LSTM layer of the pronoun has the highest similarity to the

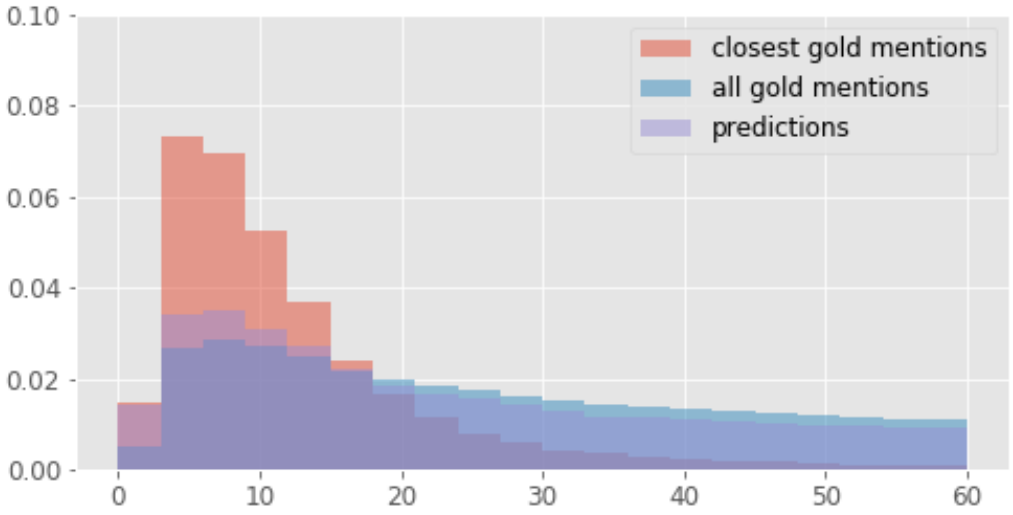

Figure 3.3: The distances between the pronoun and its gold and predicted antecedents for TransformerXL.

previous token (16.4% in the Transformer) [6]. Instead, the probe models tend to refer to entities that are further away from the target than the closest *gold* entity (74.2% cases in the case of the Transformer), suggesting that they do not rely on a simple recency bias either . This observation is confirmed when looking at the distribution of predicted antecedents and gold antecedents (Figures 3.3 and 3.2).

The great difference in performance between AWD-LSTM and TransformerXL could suggest that the latter is using different strategies compared to the former. Instead, except for the recency bias, what we find are exactly the same patterns in behavior, with a systematic 10% accuracy gap. For this reason, although we provide results for both models everywhere to show that this observation indeed holds, in this section we will mostly focus on the Transformer when commenting results.

3.3.3 Grammatical Patterns

The models clearly learn grammatical constraints related to anaphora that are well-studied in the literature and are relied upon by traditional anaphora resolution models (Sukthanker et al., 2018). First, as shown in Table 3.3, the Transformer identifies mentions (elements inside some coreference chain) in 92.6% of the cases. Moreover, it correctly learns that pronouns typically refer to nominal

[6]The attention mechanism of the Transformer gives access to a broader context and allows it to overcome the recency bias to some degree

	LSTM		Transformer	
in chain	90.2%		92.6%	
POS	Perc.	Acc	Perc.	Acc
Noun phrase	15.5	50.9	17.0	62.3
Proper noun	20.2	64.3	20.0	74.9
Pronoun	59.0	71.5	59.0	**82.6**
Other	5.3	67.3	3.0	81.6

Table 3.3: Statistics on types of mentions that the probe models refer to, for predictions that are in a coreference chain. 'Noun phrase' stands for elements that are typically within a noun phrase (note that our system points to individual tokens): Determiners, nouns, and adjectives.

elements (almost 95% identified antecedents are pronouns, proper nouns, and elements within a noun phrase headed by a common noun). Note that pronouns can also have non-nominal antecedents, although these are the minority of the annotations in OntoNotes (cf. example 4 in Figure 3.6, where *it* refers to an event). Even in the cases in which the Transformer points to elements outside of a chain (7.4%), it points to nominal elements 87% of the time (not shown in the table). The model is most accurate when referring to pronouns (82.6% accuracy), while noun phrases are the hardest category (62.3%). This is consistent with the strategies that the model learns, since it largely relies on pronominal agreement, as described below.

	LSTM		Transformer	
Pron-ant.	Perc.	Acc	Perc.	Acc
sg-sg	97.7	66.3	98.7	**76.0**
sg-pl	2.3	20.5	1.3	36.7
pl-sg	35.5	40.8	27.5	53.1
pl-pl	64.5	67.7	72.5	**72.3**

Table 3.4: The types of noun phrase antecedents the models choose, by number agreement (e.g., 'sg-pl' means 'anaphoric pronoun is singular, antecedent plural').

Second, not only do the models mostly point to nominal elements, but they also identify the morphosyntactic properties of pronouns and learn that they should agree with their antecedents in gender and number. Figure 3.4 shows the distribution of pronoun antecedents that the Transformer predicts, for the six most frequent target pronouns. Its preferred type of antecedent are pronouns of the same

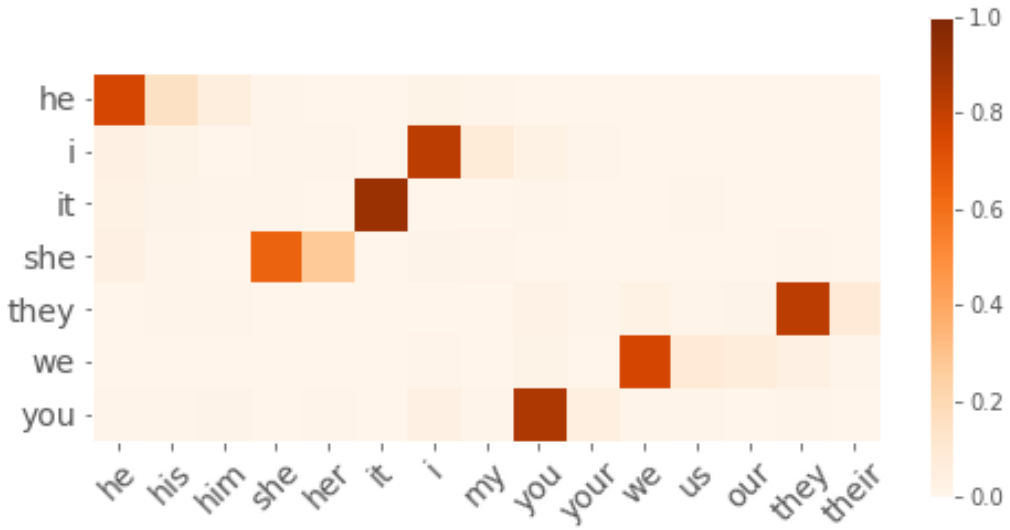

Figure 3.4: Pronominal agreement with Transformer probe model: Proportion of cases in which elements in the rows corefer with elements in the columns.

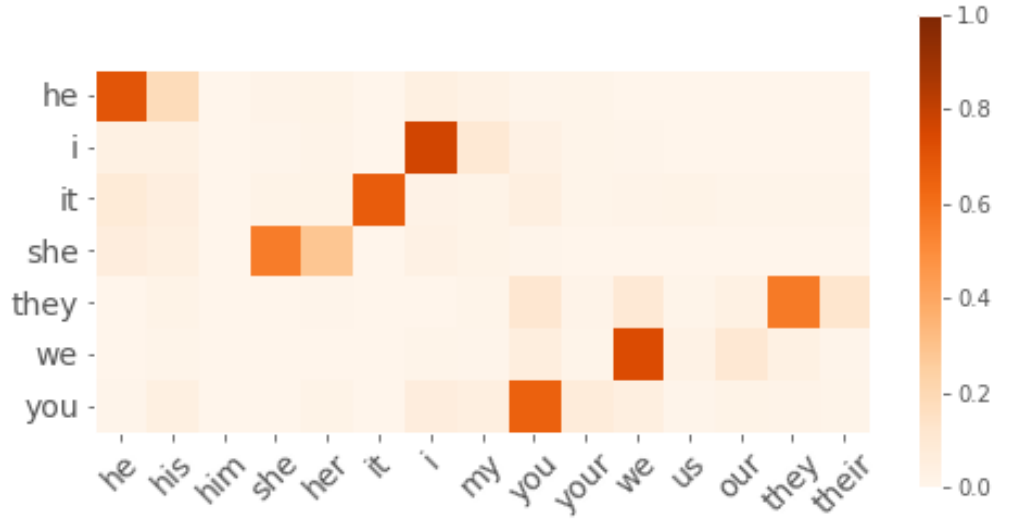

Figure 3.5: Pronominal agreement with Transformer probe model: Proportion of cases in which elements in the rows corefer with elements in the columns.

1. Why had **Mr. Korotich** been called? "I told my driver," *he* said, "that **he**
2. While **Peter** was still in the yard, a servant girl of the high *priest* came there. She saw him warming himself by the fire. She looked closely at **him**
3. The performance by more than 40 members of the Rome Philharmonic Orchestra intoxicated ***the audience*** and the musical fountain, hi-fi sound effect, fountain screen and stereographic projection brough **them**
4. Mr. Gonzalez expressed concern over *a report* that the two had been summoned to Washington by Mr. Wall last week to discuss their testimony in advance. "I think he is **trying** to improperly influence a witness, and by God I 'm not going to tolerate **it**

Figure 3.6: Difficult cases of anaphora. The target pronoun and its antecedent are in **bold**; the prediction of the model is in *italic*.

form, but it is also able to point to other pronouns agreeing in number and gender. For instance, pronoun *he* points to 3rd person, masculine, singular pronouns (mostly *he*, but also *his*, *him*) —a pattern consistent across all pronouns. We can visualize similar patterns for the LSTM model in Figure 3.5 with slightly more noise due to its worse general performance.

Figure 3.4 is restricted to pronouns; Table 3.4 shows that the models also largely follows number agreement when predicting antecedents within noun phrases (the table collapses common noun and proper noun antecedents). Given a singular pronoun, the Transformer model chooses a singular antecedent 98% of the time; given a plural pronoun, it identifies a plural antecedent in 73% of the cases.

Note that in cases of plural pronouns such as *they* it is common that the referent be a singular noun (e.g., *the audience* in example 3, Figure 3.6), reflected by the reasonable accuracy of the Transformer in pl-sg cases (53.1%).

3.4 Analysis: emergence of entity representations

The language model clearly captures grammatical properties that constrain anaphora resolution; in this section, we show that it struggles more with the semantic (referential) aspect, but it still captures it to some extent.

3.4.1 Ability to distinguish different entities

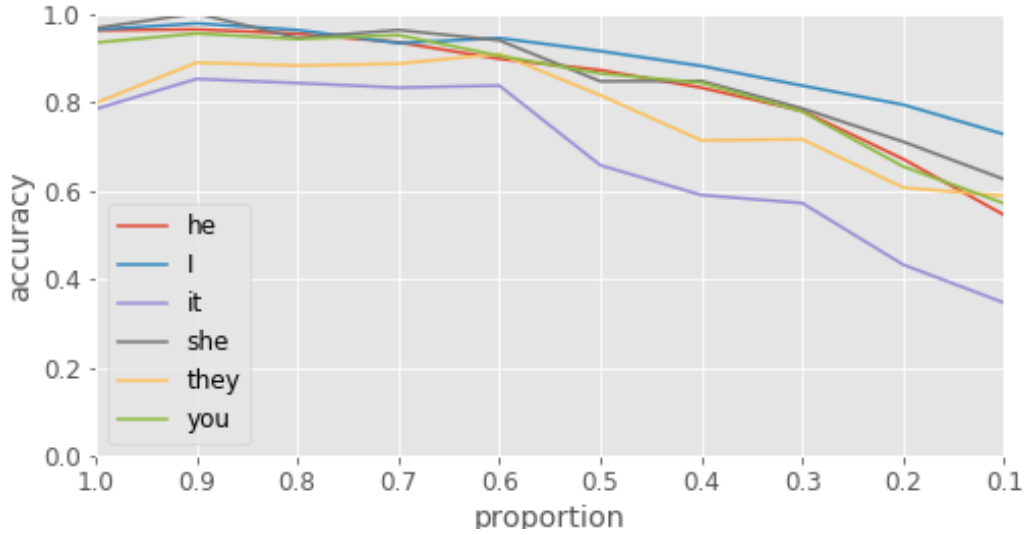

Figure 3.7: Transformer probe model: Accuracy as a function of the proportion of mentions that are antecedents (vs. distractors) in the window.

If the model were able to model entities, it should be robust to *distractors*, that is, mentions in the context that are not antecedents –in Figure 3.1, *he* and *the Central Military Commission*. Figure 3.7 shows that the accuracy for the Transformer decreases as does the proportion of gold mentions. We compute this proportion as the number of gold mentions in the 60-token window divided by the total number of mentions in the same window. When there are no distractors (gold proportion = 1), accuracy is very high, which is to be expected given that the model learnt to identify mentions in the first place (cf. previous section). The more distractors (i.e., the lower the proportion of gold mentions), the lower the accuracy; however, accuracy decreases rather gracefully. Even when there are only 10% gold mentions in the window, accuracy for most pronoun types is still around 60-80%. The exception is *it*, which is the most difficult pronoun for the model, presumably because it can refer to many kinds of antecedents. When we visualize the behaviour of the LSTM relative to the proportion of distractors (Figure 3.8, the tendencies seem to be the same as the ones for TransformerXL, even though the curves are steeper, the model being more confused with a higher number of distractors.
[7]

[7]While most personal pronouns refer to people, which are relatively homogeneous kinds of referents, *it* refers to very varied kinds of referents. Qualitative analysis suggests that the model is quite successful when *it* refers to concrete entities (*province*, *peanut*), but much less when it refers to abstract objects like propositions or events, as in example 4 of Figure 3.6 (where *it* refers to the

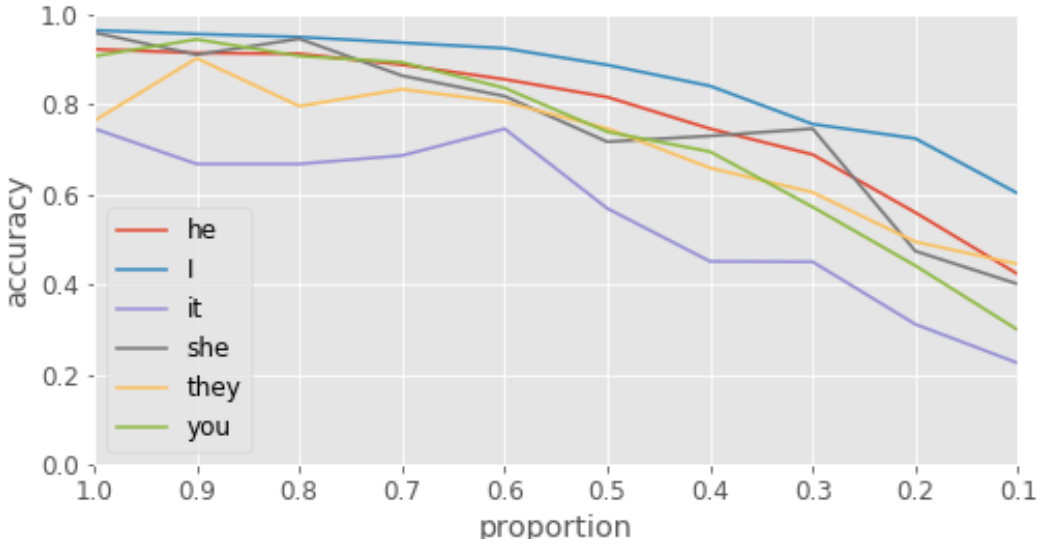

Figure 3.8: LSTM probe model: Accuracy as a function of the proportion of mentions that are antecedents (vs. distractors) in the window.

Figure 3.7 thus paints a nuanced picture: distractors confuse the model, but they do not fool it completely. Given the results in the previous section, we expect that distractors sharing morphosyntactic features will be particularly challenging. Table 3.5 confirms this, zooming in into pronominal distractors. We consider a datapoint having a pronominal distractor if one of the antecedents is a pronoun pointing to another entity.

When there are no pronominal distractors (25.9% of the test set), the accuracy of the Transformer is 81.8%; with at least one distractor, it goes down to 73.8% — clearly worse but not dramatically so. However, in cases where anaphoric pronoun and antecedent have the same gender, number, or are the same pronoun, we get much lower accuracies (48.6, 65.3, and 49.1, respectively). This suggests that that the model overly relies on morphosyntactic features and recency (see previous section).Among the hardest cases are those where two coreference chains in the window have the same pronoun (e.g. *he*) or gender (e.g. *he-his*). Most of these cases appear when the text includes reported speech (see Figure 3.6, example 1). Otherwise, there are few cases of such local ambiguity, which is presumably avoided by language speakers. However, qualitative analysis suggests that the presence of distractors is also problematic in the case of nouns, as illustrated in example 2 of Figure 3.6, where the model is presumably confused by a noun of

event of trying to improperly influence a witness). A quantitative check confirms this hypothesis: Cases in which the model fails have around 18% of verbal references, compared to less than 2% for cases in which the model is right.

Type	Perc.	L Acc.	T Acc.	Base Acc.
No distractor	25.9	74.9	**81.8**	100
Distractor(s)	74.1	61.3	73.8	32.0
= gender*	4.8	40.9	48.6	15.7
= number	37.2	55.7	65.3	26.6
= pron.	10.3	39.7	49.1	20.3

Table 3.5: Percentage of datapoints with/without pronominal distractors and accuracy of the models (LSTM - L, Transformer - T) and baseline (last column). *Excludes cases with no marked gender (like *I*, *you*).

the same gender and number as the pronoun (*priest* vs. *Peter-him*).

However, accuracy in these cases goes down but is still decent, compared to a reasonable baseline (last column in the table). For each target anaphoric pronoun, we calculate baseline accuracy as the percentage of gold pronouns in the window (pronouns that are in the same chain as the target), that is, number of gold pronouns divided by the total number of pronouns in the window. Then we calculate the average of this accuracy over the respective subset (no distractors / distractors / same gender, etc.). The baseline when there are no distractors is by definition 100%; when there are distractors, it ranges between 15.7 and 32%. All model accuracies are well above this baseline.

The results thus suggest that the models are able to distinguish mentions of different entities locally to some extent, although they are far worse at this than at capturing morphosyntactic features. In the following subsection, we provide further support for this interpretation at a more global level.

3.4.2 Clustering of mentions into entities

Our last piece of analysis looks at whole documents. We aim at testing whether the hidden representations of the language models contain information that can help distinguish mentions of the same entity from mentions of some other entity, even if they are of the same form; for instance, a pronoun *she* referring to two different women. We use coreference chains to identify the tokens referring to the same entity, and train a probe model to determine when two pronouns are referring to the same entity, that is, whether they are part of the same coreference chain in a document. In the previous probe task, where the model was trained to find a correct local antecedent, the model could use cues such as linear distance and syntactic relations; here it should rely on more persistent entity-related features in

the hidden representations.

Experimental Setup. We focus on pronouns because they cannot be disambiguated on the basis of lexical features. We use the same train/test partition as in the first probe task. For each datapoint, we have two pronouns: **x** and **y**, which can either come from the same chain, or not. Again, we take each pronoun to be represented by the last hidden layer representation of the language model (Eq. (3.1)): h_x and h_y. We call this representation *unsupervised*, and will compare it to the supervised one, obtained as follows.

Similarly to the previous probe task, the embeddings are transformed through a learnt linear transformation to a 400-dimensional vector to extract features relevant for the entity identification task (Eqs. (3.6) and (3.7)). We take the cosine between the transformed representations as the similarity between the two pronouns.

We take as positive datapoints pairs containining two pronouns belonging to the same chain, as negative datapoints two pronouns from two different chains. During training, for each document, we extract all positive pairs and then randomly select the same number of negative pairs. The model optimises max-margin loss on these datapoints (Eq. (3.8), where x and y belong to the same chain and x' and y' belong to two different chains).

$$o_x = W * h_x + b \tag{3.6}$$

$$o_y = W * h_y + b \tag{3.7}$$

$$L = 1 - cos(o_x, o_y) + cos(o_{x'}, o_{y'}) \tag{3.8}$$

To evaluate the distance metric learnt by the model we use the silhouette coefficient (Rousseeuw, 1987), which is commonly used for intrinsic clustering evaluation. The silhouette coefficient for each pronoun **x** is defined as in Eq. (3.9), where a is the mean distance between **x** and all other items in the same chain, and b is the mean distance between **x** and all other items in the closest chain (measured in the learnt space, not in terms of linear distance). Its range is $[-1, 1]$, with 1 corresponding to the pronoun being much closer to the other pronouns in its chain, 0 being borderline (equally close to the two compared chains), and -1 being much closer to the pronouns in the other chain. The average silhouette coefficient is used as an overall measure of clustering quality. A score below 0.25 is usually deemed a null result Kaufman and Rousseeuw (1990).

$$s = \frac{b - a}{max(a, b)} \tag{3.9}$$

The probe model is trained for 50 epochs, keeping the model at the best validation epoch, i.e., where the silhouette score over the validation data is highest.

In addition to the trained probe model, we provide the results on global entity discrimination for the unsupervised baseline which computes the cosine similarity between the non-transformed hidden representations of the language models, similarly to the first probe task.

Results. All the obtained values are well below 0.25. Table 3.6 contains the results for all the datapoints as well as divided into easy and difficult documents. In easy documents, all the chains have different pronouns, so they can be distinguished by the token form only. Difficult documents contain confusable chains, that is, there are at least two different chains which share the same pronoun. Coefficients are a bit higher for easy documents, but still very low, and, for complex documents, they are virtually zero. Moreover, the supervised models performs marginally better than the cosine baselines, but clearly do not learn any reliable information.

		LSTM		Transformer	
	N	unsup	sup	unsup	sup
all	1142	-0.09	0.02	-0.08	0.03
easy	194	0.12	0.14	0.13	0.16
diff	948	-0.13	-0.007	-0.13	0.01

Table 3.6: Results for the second probe task (average silhouette coefficient).

Indeed, the average distances within and across chains seem to confirm these results. If models were capturing global entity-related properties in their mention representations, we would expect pronouns with the same form but in different chains to be further away than pronouns (of any form) that belong to the same chain; instead, they are at the same distance (average cosines of 0.75 / 0.76 for Transformer, 0.74 / 0.73 for LSTM, respectively).

We conclude that the models' sensitivity to whether two identical pronouns belong to the same chain or not only shows if linear distance is factored out. If it is not, as in the current experiment, the models fail completely at distinguishing entities. This is because, with linear distance, the similarity in the entity-centered representation space shrinks very fast; same-chain mentions that are further away have lower average similarities than different-chain mentions that are nearby.

Following these findings, we analyze the results of the experiment relative to linear distance. Figure 3.9 plots the similarities between positive and negative pairs (solid and dashed lines, respectively) for the two analyzed language models,

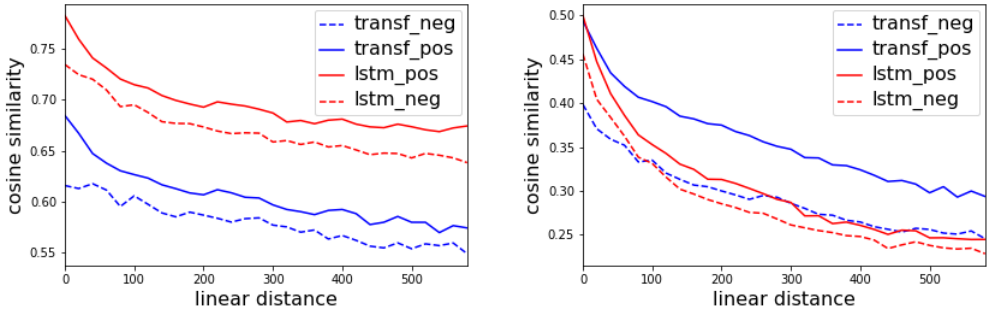

Figure 3.9: Linear distance in the discourse vs. cosine distance, for all the mention pairs with the same token pronoun. Distances averaged within bins of 20 tokens. Left: unsupervised, right: supervised.

compared to linear distance in the text. The left graph corresponds to unsupervised similarities, the right graph to supervised similarities. To control for token form effect, we only include data with the same pronoun pairs in this graph. Three results stand out. First, despite training with a global objective, with no linear information, similarities are negatively correlated with linear distance in text. This is consistent with the tendency of the unsupervised cosine baseline of pointing to the closest token (see Section 3.3).

The second result is that, crucially, after controlling both for distance and for pronoun form, similarities are systematically higher for coreferring pronoun pairs than for non-coreferring ones. Thus, some properties make their way into the hidden representations (and the probe model) that make coreferring mentions distinct from non-correferring mentions —modulo distance.

Finally, the third main result is that the supervised model is able to extract discriminating information from the hidden layers to a much larger extent in the Transformer than in the LSTM (cf. distance between blue and red lines, respectively). We interpret this to mean that such information is encoded to a larger extent in the Transformer. Also note that the supervised LSTM model is more sensitive to linear distance than any of the other representations (cf. the steeper curves between 0-100 token distances). As we signaled in the previous section, LSTM is more prone to recency biases, and it looks like global representations contain less entity-related information than in the case of the Transformer, such that the supervised model defaults to recency. We conclude from this that the Transformer accounts for semantico-referential aspects better than the LSTM.

Overall, the results suggest that token form and proximity in text remain the main properties encoded in the hidden states of entity mentions, but other properties that discriminate between coreferring and non-coreferring mentions are

present to some extent, allowing for partial discrimination.

3.5 Conclusion

Previous work has provided robust evidence that language models capture grammatical information without being explicitly trained to do so (Linzen et al., 2016; Gulordava et al., 2018b). In this chapter, we have analyzed to what extent they learn referential aspects of language, focusing on anaphora. We have tested two models representative of the prevailing architectures (Transformer, LSTM).

We find that the two models behave similarly, but the Transformer performs consistently better (around 10% higher accuracy in the probe tasks).[8]

As expected, our results show that language models capture local anaphora patterns: Based on the information in the hidden layers, a simple linear transformation learns to link pronouns to other pronouns or noun phrases, and to do so largely respecting agreement constraints in gender and number. On the other hand, models get confused when there are other mentions in the context, especially if they match in some morphosyntactic feature, but less than could be expected.

Although it is much harder for models to induce a more global notion of entity, models seem to encode slightly entity-specific information. They show some limited ability to distinguish mentions that have the same form but are in different coreference chains, though hampered by their heavy recency bias. The recency bias affects LSTMs more, but is also found in Transformers, consistent with previous work on syntax (van Schijndel et al., 2019).

Our results thus suggest that language models provide representations that capture very well contextual patterns, like gender and number. Due to the rich general linguistic representations provided by language models and the ability to encode contextual patterns, we consider them as a good base component for the entity model that we aim to develop, while the rest of the model can focus on patterns like extra-linguistic information or global semantic representations of entities which are not captured by the standard language models.

[8] With the caveat that the model we tested is slightly bigger than its LSTM counterpart.

Chapter 4

AN ENTITY CENTRIC NEURAL NETWORK FOR CHARACTER IDENTIFICATION

4.1 Introduction

Following the previous analysis studies, in this chapter, we aim at developing a model which blends together the lexical information with the contextual one (both linguistic and extra-linguistic) and develops meaningful entity representations that encode specific characteristics of entities. In order to achieve our modelling goal, we use Word2Vec embeddings for strong lexical representations of our linguistic input. Then these embeddings are connected to a bidirectional LSTM with the role of encoding the contextual information. Since the decoder attention proved very powerful in identifying and extracting discrete features in Chapter 2, we use this mechanism to query an external entity module regarding an entity represented by the output of the LSTM.

We build on a hypothesis in recent work on referential tasks such as coreference resolution and entity linking (Haghighi and Klein, 2010; Clark and Manning, 2016; Henaff et al., 2017; Clark et al., 2018), namely, that encouraging models to learn and use entity representations will help them better carry out referential tasks. To illustrate, creating an entity representation with the relevant information upon reading *a woman* should make it easier to resolve a pronoun mention like *she*.[1] In the mentioned work, several models have been proposed that incorporate an explicit bias towards entity representations. Such **entity-centric** models have shown empirical success, but we still know little about what it is that they

[1]Note the analogy with traditional models in formal linguistics like Discourse Representation Theory Kamp and Reyle (2013).

JOEY TRIBBIANI (183):
"... *see Ross, because I think you love her* ."
335 183 335 306

Figure 4.1: Character identification: example.

effectively learn to model.

In this chapter, we propose two different types of modules that can facilitate a better encoding of entities. First, we develop a static entity library, where we have a matrix with each row representing the information about an entity. This structure resembles the memory network (Sukhbaatar et al., 2015) which is one of the first developments of models with external modules. Second, we implement a dynamic entity library inspired by EntNet (Henaff et al., 2016). In this case, each entity is associated with a key vector and a value vector. The key vector should behave as an identifier, and it should encode more global characteristics of the entity. On the other hand, the value vector should encode more contextual information about each entity.

We train the developed models on the task of character identification on multi-party dialogue as posed in SemEval 2018 Task 4 Choi and Chen (2018).[2] Models are given dialogues from the TV show *Friends* and asked to link entity mentions (nominal expressions like *I*, *she* or *the woman*) to the characters to which they refer in each case. Figure 4.1 shows an example, where the mentions *Ross* and *you* are linked to entity 335, mention *I* to entity 183, etc. Since the TV series revolves around a set of entities that recur over many scenes and episodes, it is a good benchmark to analyze whether entity-centric models learn and use entity representations for referential tasks. Even though the results are promising on the task of character identification, this is not reflected by good entity representations, which we reveal through representation analysis and through a new challenge set that probes for entity information. [3]

Our contributions are three-fold: First, we propose two entity-centric models and show that they do better on lower frequency entities (a significant challenge for current data-hungry models) than a counterpart model that is not entity-centric, with the same model size. Second, through analysis, we provide insights into how they achieve these improvements, and argue that making models entity-centric fosters architectural decisions that result in good inductive biases. Third, we create a dataset and task to evaluate the models' ability to encode entity information such as gender, and show that models fail at it. More generally, we underscore in this

[2] https://competitions.codalab.org/competitions/17310.

[3] Source code for our model, the training procedure and the new dataset is published on https://github.com/amore-upf/analysis-entity-centric-nns.

chapter the need for the analysis of model behaviour, not only through ablation studies, but also through the targeted probing of model representations (Linzen et al., 2016; Conneau et al., 2018b).

4.2 Related Work

Modeling. Various memory architectures have been proposed that are not specifically for entity-centric models, but could in principle be employed in them: neural turing machines (Graves et al., 2014), memory networks (Sukhbaatar et al., 2015) Stack neural networks (Joulin and Mikolov, 2015) RelNet (Bansal et al., 2017) and EntNet (Henaff et al., 2016). We base our first model development on the memory network design due to its simplicity. In our second proposed architecture, we take inspiration from EntNet because of the idea of splitting the information in keys and values which should facilitate a more distributed encoding of contextual and global patterns.

We show that our adaptations yield good results and provide a closer analysis of their behavior.

Tasks. The task of entity linking has been formalized as resolving entity mentions to referential entity entries in a knowledge repository, mostly Wikipedia (Bunescu and Paşca, 2006; Mihalcea and Csomai, 2007 and much subsequent work; for recent approaches see (Francis-Landau et al., 2016; Chen et al., 2018). In the present entity linking task, only a list of entities is given, without associated encyclopedic entries, and information about the entities needs to be acquired from scratch through the task; note the analogy to how a human audience might get familiar with the TV show characters by watching it. Moreover, it addresses multiparty dialogue (as opposed to, typically, narrative text), where speaker information is crucial. A task closely related to entity linking is *coreference resolution*, i.e., predicting which portions of a text refer to the same entity (e.g., *Marie Curie* and *the scientist*). This typically requires clustering mentions that refer to the same entity (Pradhan et al., 2011a). Mention clusters essentially correspond to entities, and recent work on coreference and language modeling has started exploiting an explicit notion of entity (Haghighi and Klein, 2010; Clark and Manning, 2016; Yang et al., 2016). Previous work both on entity linking and on coreference resolution (cited above, as well as Wiseman et al., 2016) often presents more complex models that incorporate e.g. hand-engineered features. In contrast, we keep our underlying model basic since we want to systematically analyze how certain architectural decisions affect performance. For the same reason we deviate from previous work to entity linking that uses a specialized coreference resolution module (e.g., Chen et al., 2017).

Analysis of Neural Network Models. Our work joins a recent strand in NLP that systematically analyzes what different neural network models learn about language (Linzen et al., 2016; Kádár et al., 2017; Conneau et al., 2018b; Gulordava et al., 2018c; Nematzadeh et al., 2018, a.o.). This work, like ours, has yielded both positive and negative results: There is evidence that they learn complex linguistic phenomena of morphological and syntactic nature, like long distance agreement (Gulordava et al., 2018c; Giulianelli et al., 2018b), but less evidence that they learn how language relates to situations; for instance, Nematzadeh et al. (2018) show that memory-augmented neural models fail on tasks that require keeping track of inconsistent states of the world.

4.3 Models

We approach character identification as a classification task, and compare a baseline LSTM (Hochreiter and Schmidhuber, 1997) with two models that enrich the LSTM with a memory module designed to learn and use entity representations. LSTMs are the workhorse for text processing, and thus a good baseline to assess the contribution of this module. The LSTM processes text of dialogue scenes one token at a time, and the output is a probability distribution over the entities (the set of entity IDs are given).

4.3.1 Baseline: BILSTM

The BILSTM model is depicted in Figure 4.2. It is a standard bidirectional LSTM Graves et al. (2005), with the difference with most uses of LSTMs in NLP that we incorporate speaker information in addition to the linguistic content of utterances.

The model is given chunks of dialogue. At each time step i, one-hot vectors for token $\mathbf{t}_i$ and speaker entities $\mathbf{s}_i$ are embedded via two distinct matrices $\mathbf{W}_t$ and $\mathbf{W}_e$ and concatenated to form a vector $\mathbf{x}_i$ (Eq. 4.1, where $\|$ denotes concatenation; note that in case of multiple simultaneous speakers S_i their embeddings are summed).

$$\mathbf{x}_i = \mathbf{W}_t\ \mathbf{t}_i \parallel \sum_{\mathbf{s} \in S_i} \mathbf{W}_e\ \mathbf{s} \tag{4.1}$$

The vector $\mathbf{x}_i$ is fed through the nonlinear activation function tanh and input to a bidirectional LSTM. The hidden state $\overrightarrow{\mathbf{h}}_i$ of a *uni*directional LSTM for the i^{th} input is recursively defined as a combination of that input with the LSTM's previous hidden state $\overrightarrow{\mathbf{h}}_{i-1}$. For a *bi*directional LSTM, the hidden state $\mathbf{h}_i$ is the concatenation of the hidden states $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ of two unidirectional LSTMs which process

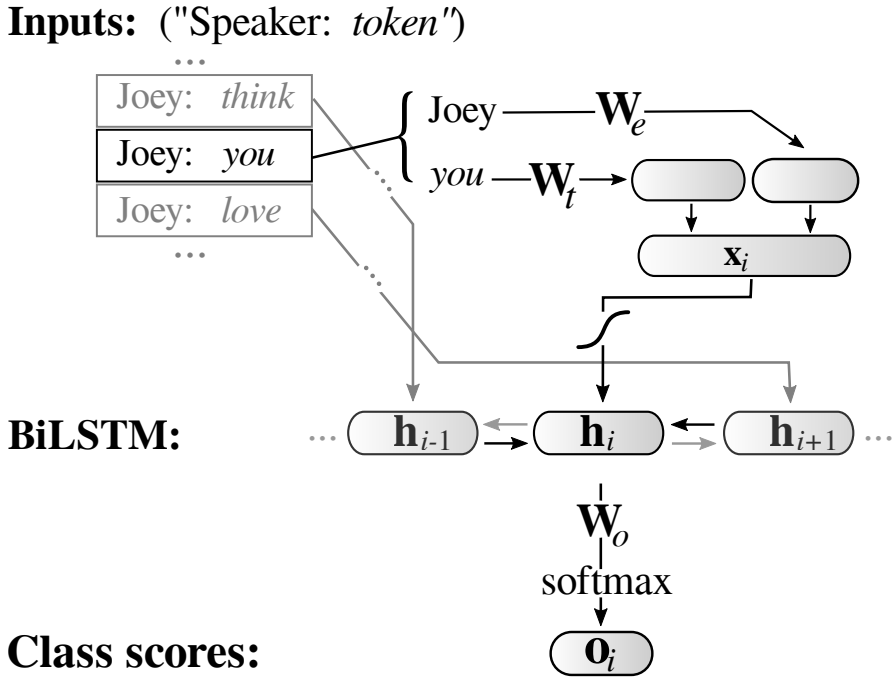

Figure 4.2: BILSTM applied to "...think you love..." as spoken by Joey (from Figure 3.1), outputting class scores for mention "you" (bias b_o not depicted).

the data in opposite directions (Eqs. 4.2-4.4).

$$\overrightarrow{\mathbf{h}_i} = \text{LSTM}(\tanh(\mathbf{x}_i), \overrightarrow{\mathbf{h}}_{i-1}) \tag{4.2}$$

$$\overleftarrow{\mathbf{h}_i} = \text{LSTM}(\tanh(\mathbf{x}_i), \overleftarrow{\mathbf{h}}_{i+1}) \tag{4.3}$$

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}_i} \parallel \overleftarrow{\mathbf{h}_i} \tag{4.4}$$

For every entity mention $\mathbf{t}_i$ (i.e., every token[4] that is tagged as referring to an entity), we obtain a distribution over all entities, $\mathbf{o}_i \in [0,1]^{1\times N}$, by applying a linear transformation to its hidden state $\mathbf{h}_i$ (Eq. 4.5), and feeding the resulting $\mathbf{g}_i$ to a softmax classifier (Eq. 4.6).

$$\mathbf{g}_i = \mathbf{W}_o\, \mathbf{h}_i + \mathbf{b}_o \tag{4.5}$$

$$\mathbf{o}_i = \text{softmax}(\mathbf{g}_i) \tag{4.6}$$

Eq. 4.5 is where the other models will diverge.

[4]For multi-word mentions this is done only for the last token in the mention.

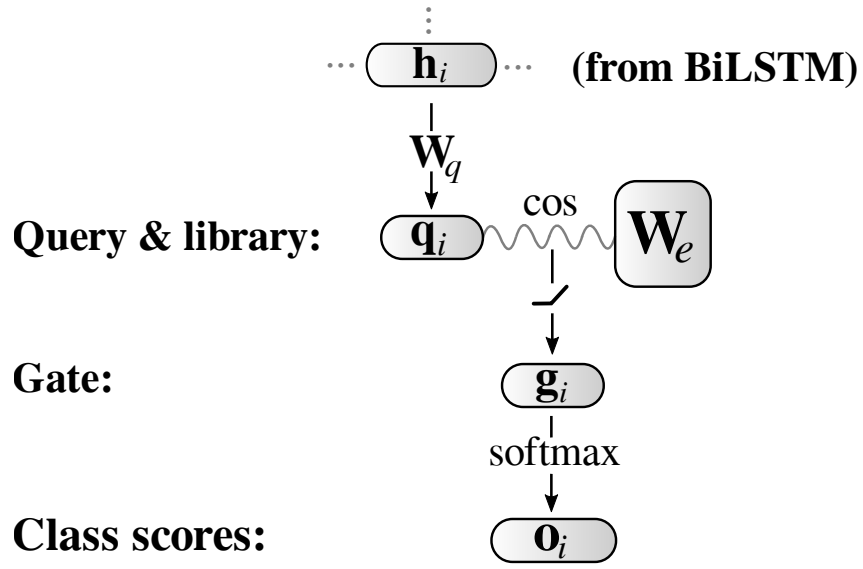

Figure 4.3: EntLib; everything before h_i, omitted here, is the same as in Figure 4.2.

4.3.2 EntLib (Static Memory)

The EntLib model (Figure 4.3) adds a simple memory module that is expected to represent entities because its vectors are tied to the output classes. We call this memory 'static entity library', since it is updated only during training, after which it remains fixed.

Where biLSTM maps the hidden state $\mathbf{h}_i$ to class scores $\mathbf{o}_i$ with a single transformation (plus softmax), EntLib instead takes two steps: It first transforms $\mathbf{h}_i$ into a 'query' vector $\mathbf{q}_i$ (Eq. 4.7) that it will then use to query the entity library.

As we will see, this mechanism helps dividing the labor between representing the context (hidden layer) and doing the prediction task (query layer).

$$\mathbf{q}_i = \mathbf{W}_q \, \mathbf{h}_i + \mathbf{b}_\mathbf{q} \tag{4.7}$$

A weight matrix $\mathbf{W}_e$ is used as the entity library, which is the same as the speaker embedding in Eq. 4.1: the query vector $\mathbf{q}_i \in \mathbb{R}^{1\times k}$ is compared to each vector in $\mathbf{W}_e$ (cosine), and a *gate* vector $\mathbf{g}_i$ is obtained by applying the ReLU function to the cosine similarity scores (Eq. 4.8).

Thus, the query extracted from the LSTM's hidden state is used as a soft pointer over the model's representation of the entities. This mechanism is similar to the decoder attention analyzed in Chapter 2, which proved to be a simple and good mechanism in direct feature detection and extraction.

$$\mathbf{g}_i = \text{ReLU}(\cos(\mathbf{W}_e, \mathbf{q}_i)) \tag{4.8}$$

As before, a softmax over $\mathbf{g}_i$ then yields the distribution over entities (Eq. 4.6). So, in the ENTLIB model Eqs. 4.7 and 4.8 together take the place of Eq. 4.5 in the BILSTM model.

We implemented an earlier version of the EntLib model as a participating model to the SemEval competition where the task of character identification was introduced. The original model did not do parameter sharing between speakers and referents, but used two distinct weight matrices. We will refer to this model as SemEval-1st in Section 4.4.

Note that the contents of the entity library in ENTLIB do not change during forward propagation of activations, but only during backpropagation of errors, i.e., during training, when the weights of $\mathbf{W}_e$ are updated. If anything, they will encode permanent properties of entities, not properties that change within a scene or between scenes or episodes, which should be useful for reference. The next model attempts to overcome this limitation.

4.3.3 ENTNET (Dynamic Memory)

ENTNET is a model inspired by the structure proposed in *Recurrent Entity Networks* (Henaff et al., 2017, Figure 4.4) and we adapt it to the task. Instead of representing each entity by a single vector, as in ENTLIB, here each entity is represented jointly by a context-invariant or 'static' *key* and a context-dependent or 'dynamic' *value*. For the keys the entity embedding $\mathbf{W}_e$ is used, just like the entity library of ENTLIB. But the values $\mathbf{V}_i$ can be dynamically updated throughout a scene.

As before, an entity query $\mathbf{q}_i$ is first obtained from the BILSTM (Eq. 4.7). Then, ENTNET computes gate values $\mathbf{g}_i$ by estimating the query's similarity to both keys and values, as in Eq. 4.9 (replacing Eq. 4.8 of ENTLIB). [5] Output scores $\mathbf{o}_i$ are computed as in the previous models (Eq. 4.6).

$$\mathbf{g}_i = \text{ReLU}(\cos(\mathbf{W}_e, \mathbf{q}_i) + \cos(\mathbf{V}_i, \mathbf{q}_i)) \tag{4.9}$$

The values $\mathbf{V}_i$ are initialized at the start of every scene ($i = 0$) as being identical to the keys ($\mathbf{V}_0 = \mathbf{W}_e$). After processing the i^{th} token, new information can be

[5] Two small changes with respect to the original model (motivated by empirical results in the hyperparameter search) are that we compute the gate using cosine similarity instead of dot product, and the obtained similarities are fed through a ReLU nonlinearity instead of sigmoid.

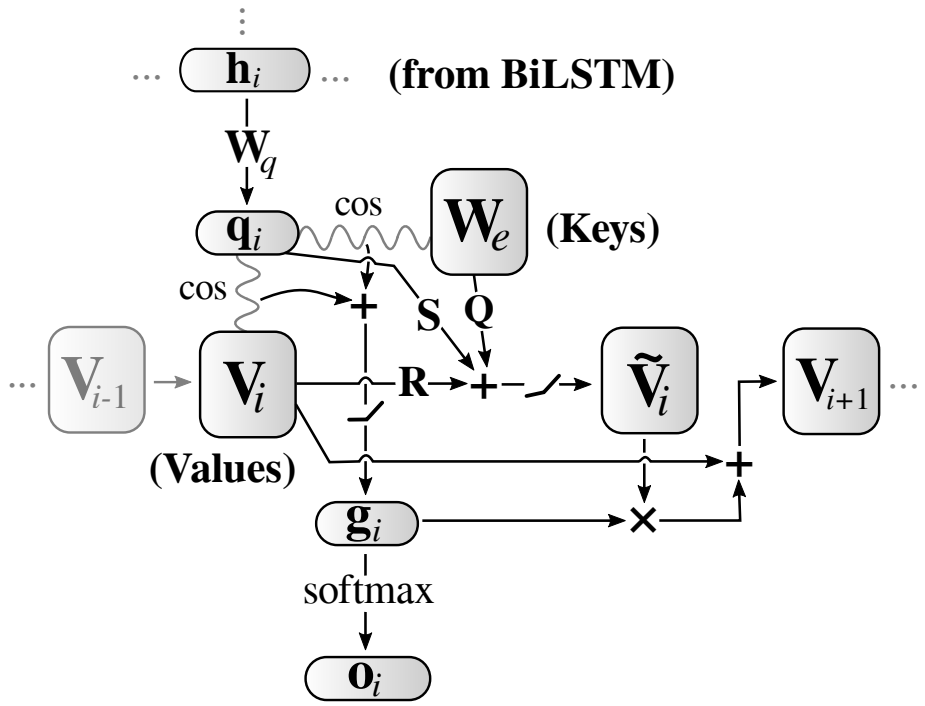

Figure 4.4: ENTNET; everything before h_i, omitted here, is the same as in Figure 4.2.

added to the values. Eq. 4.10 computes this new information $\tilde{\mathbf{V}}_{i,j}$, for the j^{th} entity, where **Q**, **R** and **S** are learned linear transformations and PReLU denotes the parameterized rectified linear unit He et al. (2015):

$$\tilde{\mathbf{V}}_{i,j} = \text{PReLU}(\mathbf{Q}\mathbf{W}_{ej} + \mathbf{R}\mathbf{V}_{i,j} + \mathbf{S}\mathbf{q}_i) \tag{4.10}$$

This information $\tilde{\mathbf{V}}_{i,j}$, multiplied by the respective gate $\mathbf{g}_{i,j}$, is added to the values to be used when processing the next $(i+1^{\text{th}})$ token (Eq. 4.11), and the result is normalized (Eq. 4.12):

$$\mathbf{V}_{i+1,j} = \mathbf{V}_j + \mathbf{g}_{i,j} * \tilde{\mathbf{V}}_{i,j} \tag{4.11}$$

$$\mathbf{V}_{i+1,j} = \frac{\mathbf{V}_{i+1,j}}{\|\mathbf{V}_{i+1,j}\|} \tag{4.12}$$

Our adaptation of the Recurrent Entity Network involves two changes. First, we use a biLSTM to process the linguistic utterance, while Henaff et al. (2017) used a simple multiplicative mask (we have natural dialogue, while their main evaluation was on bAbI, a synthetic dataset). Second, in the original model the

gates were used to retrieve and output information about the query, whereas we use them directly as output scores because our task is referential. This also allows us to tie the keys to the characters of the Friends series as in the previous model, and thus have them represent entities (in the original model, the keys represented entity types, not instances).

4.3.4 Hyperparameter search

Besides the LSTM parameters, we optimize the token embeddings $\mathbf{W}_t$, the entity/speaker embeddings $\mathbf{W}_e$, as well as $\mathbf{W}_o$, $\mathbf{W}_q$, and their corresponding biases, where applicable. We used five-fold cross-validation with early stopping based on the validation score. We found that most hyperparameters could be safely fixed the same way for all three types. Specifically, our final models were all trained in batch mode using the Adam optimizer Kingma and Ba (2014), with each batch covering 25 scenes given to the model in chunks of 750 tokens. The token embeddings ($\mathbf{W}_t$) are initialized with the 300-dimensional word2vec vectors, $\mathbf{h}_i$ is set to 500 units, and entity (or speaker) embeddings ($\mathbf{W}_e$) to k = 150 units.With this hyperparameter setting, ENTLIB has fewer parameters than BILSTM: the linear map $\mathbf{W}_o$ of the latter (500×401) is replaced by the query extractor $\mathbf{W}_q$ (500×150) followed by (non-parameterized) similarity computations. This holds even if we take into account that the entity embedding $\mathbf{W}_e$ used in both models contains 274 entities that are never speakers and that are, hence, used by ENTLIB but not by BILSTM.

Our search also considered different types of activation functions in different places, with the architecture presented above, i.e., tanh before the LSTM and ReLU in the gate, robustly yielding the best results. Other settings tested—randomly initialized token embeddings, self-attention on the input layer, and a uni-directional LSTM—did not improve performance.

We then performed another random search ($>$ 200 models) over the remaining hyperparameters: learning rate (sampled from the logarithmic interval 0.001–0.05), dropout before and after LSTM (sampled from 0.0–0.3 and 0.0–0.1, respectively), weight decay (sampled from 10^{-6}–10^{-2}) and penalization, i.e., whether to decrease the relative impact of frequent entities by dividing the loss for an entity by the square root of its frequency. We report the best model of each type, i.e., BILSTM, ENTLIB, and ENTNET, after training on all the training data without cross-validation for 20, 40 and 80 epochs respectively (numbers selected based on tendencies in training histories). These models had the following parameters:

	BILSTM	ENTLIB	ENTNET
learning rate:	0.0080	0.0011	0.0014
dropout pre	0.2	0.2	0.0
dropout post:	0.0	0.02	0.08
weight decay:	1.8e-6	4.3e-6	1.0e-5
penalization:	no	yes	yes

Table 4.1: Hyperparameter setup for the proposed models

	Train	Test
Entities	372	106
Mentions	13,280	2,429
Scenes	374	74
Episodes	47	40

Table 4.2: Summary statistics of the SemEval 2018 Task 4 dataset. The union of entities in the training and test sets is 401.

4.4 Character Identification

The training and test data for the task span the first two seasons of *Friends*, divided into scenes and episodes, which were in turn divided into utterances (and tokens) annotated with speaker identity.[6] The set of all possible entities to refer to is given, as well as the set of mentions to resolve. Only the dialogues and speaker information are available (e.g., no video or descriptive text). Indeed, one of the most interesting aspects of the SemEval data is the fact that it is dialogue (even if scripted), which allows us to explore the role of speaker information, one of the aspects of the extralinguistic context of utterance that is crucial for reference.

We additionally used the publicly available 300-dimensional word vectors that were pre-trained on a Google News corpus with the word2vec Skip-gram model Mikolov et al. (2013a) to represent the input tokens. Entity (speaker/referent) embeddings were randomly initialized.

We train the models with backpropagation, using the standard negative log-likelihood loss function. For each of the three model architectures we performed a random search ($>$ 1500 models) over the hyperparameters using cross-validation as presented in the previous section, and report the results of the best settings after retraining without cross-validation. The findings we report are representative of the model populations.

[6]The dataset also includes automatic linguistic annotations, e.g., PoS tags, which our models do not use.

		all (78)		main (7)	
models	#par	F_1	Acc	F_1	Acc
SemEv-1st	-	41.1	74.7	79.4	77.2
SemEv-2nd	-	13.5	68.6	83.4	82.1
BILSTM	3.4M	34.4	74.6	**85.0**	83.5
ENTLIB	3.3M	49.6*	**77.6***	84.9	**84.2**
ENTNET	3.4M	**52.5***	77.5*	84.8	83.9

Table 4.3: Model parameters and results on the character identification task. First block: top systems at SemEval 2018. Results in the second block marked with * are statistically significantly better than BILSTM at $p < 0.001$ (approximate randomization tests, Noreen, 1989).

Results. We follow the evaluation defined in the SemEval task (Choi and Chen, 2018). Metrics are macro-average F_1-score (which computes the F_1-score for each entity separately and then averages these over all entities) and accuracy, in two conditions: *All entities*, with 78 classes (77 for entities that are mentioned in both training and test set of the SemEval Task, and one grouping all others), and *main entities*, with 7 classes (6 for the main characters and one for all the others). Macro-average F_1-score on all entities, the most stringent, was the criterion to define the leaderboard.

Table 4.3 gives our results in the two evaluations, comparing the models described in Section 4.3 to the best performing models in the SemEval 2018 Task 4 competition: the first iteration of our EntLib as "SemEv-1st", and the model proposed by Park et al. (2018) as "SemEv-2nd". Our proposed modules outperform the previous developed models and the models containing one of the proposed external modules improve over the base model. This signals that the addition of a structure designed for encoding entity information is beneficial and we plan to conduct further analysis in order to reveal what are the phenomena which are captured better in the newly developed models.

All models perform on a par on main entities, but entity-centric models outperform BILSTM by a substantial margin when all characters are to be predicted (the difference between ENTLIB and ENTNET is not significant).

The architectures of ENTLIB and ENTNET help with lower frequency characters, while not hurting performance on main characters. Indeed, Figure 4.5 shows that the accuracy of BILSTM rapidly deteriorates for less frequent entities, whereas ENTLIB and ENTNET degrade more gracefully. Deep learning approaches are data-hungry, and entity mentions follow the Zipfian distribution typical of language, with very few high frequency and many lower-frequency items, such that this is a welcome result.

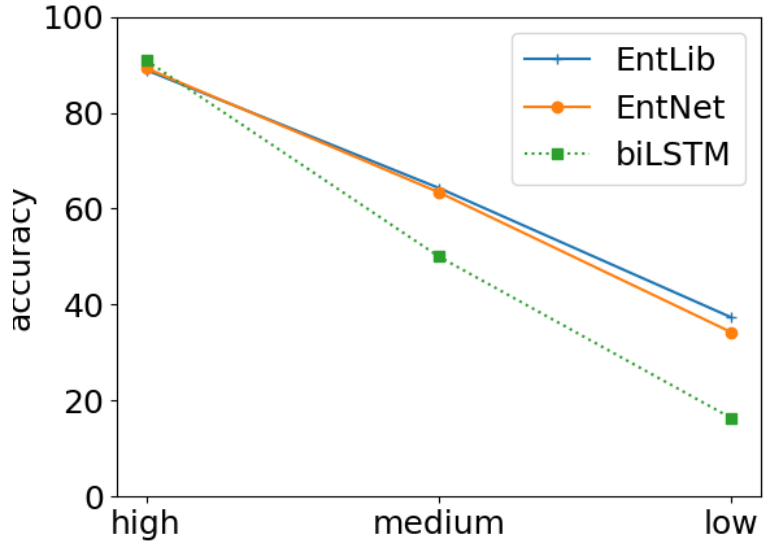

Figure 4.5: Accuracy on entities with high (>1000), medium (20–1000), and low (<20) frequency.

Moreover, these improvements do not come at the cost of model complexity in terms of number of parameters, since all models have roughly the same number of parameters ($3.3 - 3.4$ million).

Given these results and the motivations for the model architectures, it would be tempting to conclude that encouraging models to learn and use entity representations helps in this referential task. However, a closer look at the models' behavior reveals a much more nuanced picture.

Figure 4.6 suggests that: (1) models are quite good at using speaker information, as the best performance is for first-person pronouns and determiners (*I*, *my*, etc.); (2) instead, models do not seem to be very good at handling other contextual information or entity-specific properties, as the worst performance is for third-person mentions and common nouns, which require both;[7] (3) ENTLIB and ENTNET behave quite similarly, with performance boosts in (1) and smaller but consistent improvements in (2). Our analyses in the next two sections confirm this picture and relate it to the models' architectures.

[7] first person: *I*, *me*, *my*, *myself*, *mine*; second person: *you*, *your*, *yourself*, *yours*; third person: *she*, *her*, *herself*, *hers*, *he*, *him*, *himself*, *his*, *it*, *itself*, *its*.

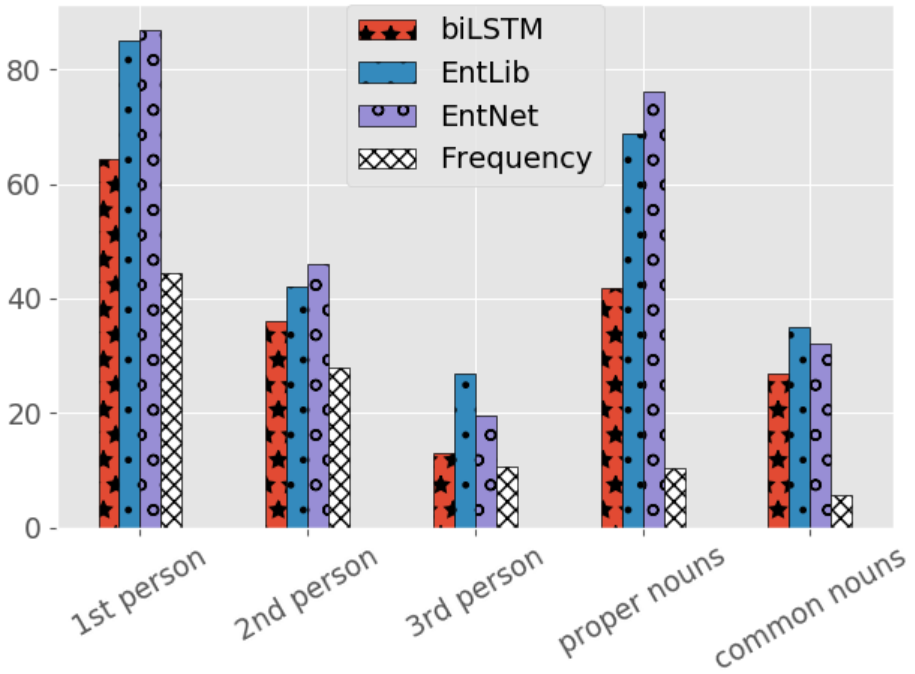

Figure 4.6: F_1-score (*all entities* condition) of the three models, per mention type, and token frequency of each mention type.

4.5 Analysis: Architecture

We examine how the entity-centric architectures improve over the BILSTM baseline on the reference task, then move to entity representations (Section 4.6).

Shared speaker/referent representation. We found that an important advantage of the entity-centric models, in particular for handling low-frequency entities, lies in the integrated representations they enable of entities both in their role of speakers and in their role of referents. This explains the boost in first-person pronoun and proper noun mentions, as follows.

Recall that the integrated representation is achieved by parameter sharing, using the same weight matrix $\mathbf{W}_e$ as speaker embedding and as entity library/keys. This enables entity-centric models to learn the linguistic rule "a first-person pronoun (*I, me*, etc.) refers to the speaker" regardless of whether they have a meaningful representation of this particular entity: It is enough that speaker representations are distinct, and they are because they have been randomly initialized. In contrast, the simple BILSTM baseline needs to independently learn the mapping between speaker embedding and output entities, and so it can only learn to resolve

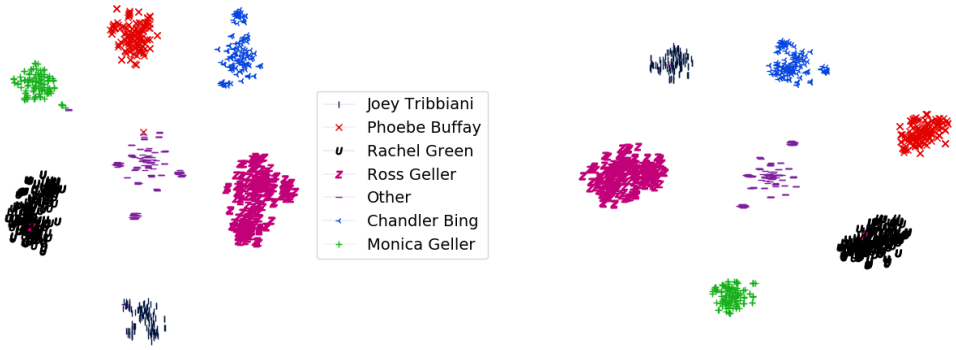

Figure 4.7: ENTLIB, 2D TSNE projections of the activations for first-person mentions in the test set, colored by the entity referred to. The mentions cluster into entities already in the hidden layer h_i (left graph; query layer q_i shown in the right graph). Best viewed in color.

model type	main	all
BILSTM	0.39	0.02
ENTLIB	0.82	0.13
ENTNET	0.92	0.16
#pairs	21	22155

Table 4.4: RSA correlation between speaker/referent embeddings W_e and token embeddings W_t of the entities' names, for main entities vs. all entities (right)

even first-person pronouns for entities for which it has enough data.

For proper nouns (character names), entity-centric models learn to align the token embeddings with the entity representations (identical to the speaker embeddings). We show this by using Representation Similarity Analysis (RSA) (Kriegeskorte et al., 2008), which measures how topologically similar two different spaces are as the Spearman correlation between the pair-wise similarities of points in each space (this is necessary because entities and tokens are in different spaces). For instance, if the two spaces are topologically similar, the relationship of entities 183 and 335 in the entity library will be analogous to the relationship between the names *Joey* and *Ross* in the token space. Table 4.4 shows the topological similarities between the two spaces, for the different model types.[8] This reveals that in entity-centric models the space of speaker/referent embeddings is

[8] As an entity's name we here take the proper noun that is most frequently used to refer to the entity in the training data. Note that for the *all entities* condition the absolute values are lower, but the space is much larger (over 22K pairs). Also note that this is an instance of slow learning; models are not encoding the fact that a proper noun like *Rachel* can refer to different people.

Figure 4.8: ENTLIB, 2D TSNE projections of the activations for mentions in the test set (excluding first-person mentions), colored by the entity referred to. While there is already some structure in the hidden layer h_i (left graph), the mentions cluster into entities much more clearly in the query q_i (right graph). Best viewed in color.

topologically very similar to the space of token embeddings restricted to the entities' names, and more so than in the BILSTM baseline. We hypothesize that entity-centric models can do the alignment better because referent (and hence speaker) embeddings are closer to the error signal, and thus backpropagation is more effective (this again helps with lower-frequency entities).

Further analysis revealed that in entity-centric models the beneficial effect of weight sharing between the speaker embedding and the entity representations (both $\mathbf{W}_e$) is actually restricted to first-person pronouns. For other expressions, having two distinct matrices yielded almost the same performance as having one (but still higher than the BILSTM, thanks to the other architectural advantage that we discuss below).

In the case of first-person pronouns, the speaker embedding given as input corresponds to the target entity. This information is already accessible in the hidden state of the LSTM. Therefore, mentions cluster into entities already at the hidden layer h_i, with no real difference with the query layer q_i (see Figure 4.7).

Advantage of query layer. The entity querying mechanism described above entails having an extra transformation after the hidden layer, with the query layer $\mathbf{q}$. Part of the improved performance of entity-centric models, compared to the BILSTM baseline, is due not to their bias towards 'entity representations' per se, but due to the presence of this extra layer. Recall that the BILSTM baseline maps the LSTM's hidden state $\mathbf{h}_i$ to output scores $\mathbf{o}_i$ with a single transformation. (Gulordava et al., 2018a) observe in the context of Language Modeling that this

BILSTM	ENTLIB		ENTNET	
$\mathbf{h}_i$	$\mathbf{h}_i$	$\mathbf{q}_i$	$\mathbf{h}_i$	$\mathbf{q}_i$
0.34	0.24	0.48	0.27	0.60

Table 4.5: Average cosine similarity of mentions with the same referent.

creates a tension between two conflicting requirements for the LSTM: keeping track of contextual information across time steps, and encoding information useful for prediction in the current timestep. The intermediate query layer $\mathbf{q}$ in entity-centric models alleviates this tension. This explains the improvements in context-dependent mentions like common nouns or second and third pronouns.

We show this effect in two ways. First, we compare the average mean similarity s of mention pairs $T_e = \{(t_k, t_{k'}) | \, t_k \rightarrow e \wedge k \neq k'\}$ referring to the same entity e in the hidden layer (Eq. 4.13) and the query layer.[9]

$$s = \frac{1}{|E|} \sum_{e \in E} \frac{1}{|T_e|} \sum_{(t_k, t_{k'}) \in T_e} \cos(h_{t_k}, h_{t_{k'}}) \tag{4.13}$$

Table 4.5 shows that, in entity-centric models, this similarity is lower in the hidden layer $\mathbf{h}_i$ than in the case of the BILSTM baseline, but in the query layer $\mathbf{q}_i$ it is instead much higher. The hidden layer thus is representing other information than referent-specific knowledge, and the query layer can be seen as extracting referent-specific information from the hidden layer. Figure 4.8 visually illustrates the division of labor between the hidden and query layers. Second, we compared the models to variants where the cosine-similarity comparison is replaced by an ordinary dot-product transformation, which converts the querying mechanism into a simple further layer. These variants performed almost as well on the reference task, albeit with a slight but consistent edge for the models using cosine similarity.

No dynamic updates in ENTNET. A surprising negative finding is that ENTNET is not using its dynamic potential on the referential task. We confirmed this in two ways. First, we tracked the values $\mathbf{V}_i$ of the entity representations and found that the pointwise difference in $\mathbf{V}_i$ at any two adjacent time steps i tended to zero. Second, we simply switched off the update mechanism during testing and did not observe any score decrease on the reference task. ENTNET is thus only using the part of the entity memory that it shares with ENTLIB, i.e., the keys $\mathbf{W_e}$, which explains their similar performance.

This finding is markedly different from Henaff et al. (2017), where for instance the BaBI tasks could be solved only by dynamically updating the entity representations. This may reflect our different language modules: since our LSTM module

[9]For the query layer, Eq. 4.13 is equivalent, with $\cos(q_{t_k}, q_{t_{k'}})$.

This person is {a/an/the} <PROPERTY> [and {a/an/the} <PROPERTY>]{0,2}.

This person is the brother of Monica Geller.
This person is a paleontologist and a man.

Figure 4.9: Patterns and examples (in italics) of the dataset for information extraction as entity linking.

already has a form of dynamic memory, unlike the simpler sentence processing module in Henaff et al. (2017), it may be that the LSTM takes this burden off of the entity module. An alternative is that it is due to differences in the datasets. We leave an empirical comparison of these potential explanations for future work, and focus in Section 4.6 on the static entity representations $\mathbf{W}_e$ that ENTNET essentially shares with ENTLIB.

4.6 Analysis: Entity Representations

The foregoing demonstrates that entity-centric architectures help in a reference task, but not that the induced representations in fact contain meaningful entity information. In this section we deploy these representations on a new dataset, showing that they do not—not even for basic information about entities such as gender.

Method. We evaluate entity representations with an information extraction task including attributes and relations, using information from an independent, unstructured knowledge base—the Friends Central Wikia.[10] To be able to use the models as is, we set up the task in terms of entity linking, asking models to solve the reference of natural language descriptions that uniquely identify an entity. For instance, given *This person is the brother of Monica Geller.*, the task is to determine that *person* refers to *Ross Geller*, based on the information in the sentence.[11] The information in the descriptions was in turn extracted from the Wikia. We do not retrain the models for this task in any way—we simply deploy them.

We linked the entities from the Friends dataset used above to the Wikia through a semi-automatic procedure that yielded 93 entities, and parsed the Wikia to extract their attributes (*gender* and *job*) and relations (e.g., *sister*, *mother-in-law*).

[10] `http://friends.wikia.com`.

[11] The referring expression is the whole DP, *This person*, but we follow the method used in the main experiments of asking for reference resolution at the head noun.

model	description	gender	job	relatives
RANDOM	1.5	50	20	16
BILSTM	0.4	-	-	-
ENTLIB	2.2	55	27	22
ENTNET	1.3	61	24	26

Table 4.6: Results on the attribute and relation prediction task: percentage accuracy for natural language descriptions, mean reciprocal rank of characters for single attributes (lower is worse).

More in detail, we performed a two-step procedure to extract all the available data for the SemEval characters. First, using simple word overlap, we automatically mapped the 401 SemEval names to the characters in the database. In a second, manual step, we corrected these mappings and added links that were not found automatically due to name alternatives, ambiguities or misspellings (e.g., SemEval *Dana* was mapped to *Dana Keystone*, and *Janitor* to *The Zoo Employee*). In total, we found 93 SemEval entities in Friends Central, and we extracted their attributes (gender and job) and their mutual relationships (relatives).

We automatically generate the natural language descriptions with a simple pattern (Figure 4.9) from combinations of properties that uniquely identify a given entity within the set of Friends characters. Also, models require inputting a speaker, so we use speaker UNKNOWN. We consider unique descriptions comprising at most 3 properties. Each property is expressed by a noun phrase, whereas the article is adapted (definite or indefinite) depending on whether that property applies to one or several entities in our data. This yields 231 unique natural language descriptions of 66 characters, created on the basis of overall 61 relation types and 56 attribute values.

Results. The results of this experiment are negative: The first column of Table 4.6 shows that models get accuracies near 0, our prosed models having a performance similar to the random baseline.

A possibility is that models do encode information in the entity representations, but it doesn't get used in this task because of how the utterance is encoded in the hidden layer, or that results are due to some quirk in the specific setup of the task.

However, we replicated the results in a setup that does not encode whole utterances but works with single attributes and relations.

Gender and job analysis. We use the same models, i.e. ENTLIB and ENTNET trained on the character identification task, and (without further training) extract

representations for the entities from them. These representations are directly obtained from the entity embedding $\mathbf{W}_e$ of each model.

For this analysis, we are given an attribute (e.g., *gender*), and all its possible values $\mathcal{V}$ (e.g., $\mathcal{V} = \{$*woman, man* $\}$).

We formulate the task as, given a character (e.g., *Rachel*), producing a ranking of the possible values in descending order of their similarity to the character, where similarity is computed by measuring the cosine of the angle between their respective vector representations in the entity space. We obtain representations of attributes values, in the same space as the entities, by inputting each attribute value as a separate utterance to the models, and extracting the corresponding entity query ($\mathbf{q}_i$). Since the models also expect a speaker for each utterance, we set the speaker to UNKNOWN.

We evaluate the rankings produced for both tasks in terms of mean reciprocal rank Craswell (2009), scoring from 0 to 1 (from worst to best) the position of the target labels in the ranking. The second and the third columns from Table 4.6 present the results. Our models generally perform poorly on the tasks, though outperforming a random baseline. Even in the case of an attribute like gender, which is crucial for the resolution of third-person pronouns, the models' results are still very close to that of the random baseline.

Relation prediction. Instead, the task of **relation prediction** is to, given a pair of characters (e.g., *Ross* and *Monica*), predict the relation R which links them (e.g., *sister, brother-in-law, nephew*; we found 24 relations that applied to at least two pairs). We approach this following the vector offset method introduced by Mikolov et al. (2013b) for semantic relations between words. This leverages on regularities in the embedding space, taking the embeddings of pairs that are connected by the same relation to having analogous spatial relations.

For two pairs of characters (a,b) and (c,d) which bear the same relation R, we assume $\mathbf{a}-\mathbf{b} \approx \mathbf{c}-\mathbf{d}$ to hold for their vector representations. For a target pair (a,b) and a relation R, we then compute the following measure:

$$\mathrm{s}_{rel}((a,b),R) = \frac{\sum_{(x,y)\in R} \cos(\mathbf{a}-\mathbf{b}, \mathbf{x}-\mathbf{y})}{|R|} \tag{4.14}$$

Equation (4.14) computes the average relational similarity between the target character pair and the exemplars of that relation (excluding the target itself), where the relational similarity is estimated as the cosine between the vector differences of the two pairs of entity representations respectively. Due to this setup, we restrict to predicting relation types that apply to at least two pairs of entities. For each target pair (a,b), we produce a rank of candidate relations in descending order of their scores s_{rel}. The last column of Table 4.6 contains the results, again above baseline but clearly very poor.

Thus, we take it to be a robust result that entity-centric models trained on the SemEval data do not learn or use entity information at a satisfactory level —at least as recoverable from language cues. This, together with the remainder of the results presented above, suggests that models rely crucially on speaker information, but hardly on information from the linguistic context. Note that 44% of the mentions in the dataset are first person, for which linguistic context is irrelevant and the models only need to recover the relevant speaker embedding to succeed. However, downsampling first-person mentions did not improve results on the other mention types. Future work should explore alternatives such as pre-training with a language modeling task, which could improve the use of context.

4.7 Conclusion

The two proposed model variations contain an external structure that was designed to capture entity information. Compared to a similar model without entity library (BiLSTM), the proposed architectures perform particularly well on rare entities, which is reflected in higher scores when we aggregate the results on all the entities. This finding is encouraging, because rare entities are especially challenging for the usual approaches in NLP due to the scarcity of information about them.

We offer the following explanation for this beneficial effect of the entity library, as a hypothesis for future work. Having an entity library requires the LSTM of our model to output some representation of the mentioned entity, as opposed to outputting class scores more or less directly as in the variant BiLSTM. Outputting a meaningful entity representation is particularly easy in the case of first-person pronouns and nominal mentions (where the beneficial effect of the entity library appears to reside; Figure 4.6): the LSTM can learn to simply forward the speaker embedding unchanged in the case of pronoun *I*, and the token embedding in the case of nominal mentions. This strategy does not discriminate between frequent and rare entities; it works for both alike.

For referring expressions that require either strong entity representations or an integration of the context, such as common nouns or third-person pronouns, all three model variants have a very poor performance. This trend indicates that training the LSTM component from scratch on this amount of data doesn't result in a good contextualization, nor strong global entity information. We will address this limitation in the following chapter, by substituting the LSTM component with a pre-trained contextualized language model.

Even though we expected a difference in performance and behaviour between the two proposed models, the model enhanced with the dynamic module converges to a similar behaviour to the one of the architecture enhanced with the static module. Therefore, our hypothesis that the value vector of the dynamic module

would encode more episodic information about entities was not met. Further research is needed to understand how to incorporate this feature in the additional module.

We furthermore analyzed the entity representations developed by the proposed models in terms of their ability to capture the properties of entities, and created a new challenge dataset that we used to probe entity representations. We showed through our analysis that, even though the models reach good performance on the character identification task, they do not yield operational entity representations, neither do they make good use of contextual information for the referential task.

More generally, we highlight in this chapter the need for model analysis to test whether the motivations for particular architectures are borne out in how the model actually behaves when it is deployed.

Chapter 5

LANGUAGE KNOWLEDGE IMPACT ON CHARACTER IDENTIFICATION

5.1 Introduction

We show in Chapter 3 that pre-trained language models encode morpho-syntactic features and local information about entities while they struggle when they need to encode semantico-referential information.

In Chapter 4, we developed an architecture with a dedicated entity library which showed promise in encoding extra-linguistic patterns, such as "the word **I** refers to the speaker" or "the word **Ross** always refers to the character Ross", while it struggled with referential expressions that require a better understanding of the context, like third-person pronouns and noun phrases.

In this chapter, we propose a unified architecture for character identification, with the aim of combining the contextualization features encoded in pre-trained language models with the extra-linguistic patterns developed by the entity library. We hypothesize that a model that is fine-tuned on the task of character identification can find a balance between the strengths previously exhibited by its components and reach a better performance than the previously developed models. Furthermore, we aim for better global entity representations, which we think can be obtained via the richer lexical representations and better contextualization provided by the pre-trained language model.

We run the experiments on the same task as in Chapter 4, followed by different analyses to obtain more insights into the knowledge encoded in these models. We expect that this will take us one step closer to successfully modelling entities.

Our hypothesis is only partially borne out. On the positive side, our proposed

model improves the performance for contextualized input, such as third-person pronouns. On the negative side, it encounters problems with second-person pronouns because it doesn't contextualize the speaker information. The pre-trained models are designed for narrative text, and they are not familiar with speaker information as input. Additionally, we show through our probing mechanisms that the model doesn't encode entity attributes or relationships, but it contains traces of gender information.

5.2 Related work

Pre-trained language models had a strong impact in the field in recent years, dominating most developments in computational linguistics.

This type of model is trained on a big amount of textual data with no need for further annotation. They are trained on general tasks such as predicting the next word (TransformerXL (Vaswani et al., 2017)) or predicting a masked word (BERT variants (Devlin et al., 2018)). While we analyzed the referential capacities of TransformerXL in Chapter 3, we decided to use BERT in the current experiments because it exhibits better performance in the majority of NLP tasks.

It has been shown that BERT can be used after this pre-training stage on a diverse set of NLP tasks, such as language understanding, question answering, common sense inference, named entity recognition (Devlin et al., 2018). For text classification tasks such as language understanding tasks (Wang et al., 2018), the pipeline is the following: after the model is fully trained on language modelling, the text input goes through the BERT component and then the representation is encoded in a special token called "**[CLS]**" which is used as input for a small classifier that solves the task. During the training stage on the new task, the information is also backpropagated in the BERT layers in order to fine-tune it to the new task. For token classification tasks like named entity recognition, the BERT output of each token is fed to the small classifier in order to predict features related to the token. The task used in this chapter is a token classification task, so we follow the second approach for fine-tuning BERT and adapting it to a new task.

Even though the initial BERT architecture is used widely in computational linguistics, there is current research that has developed the model further. First, there are variants of BERT that are designed for specific types of text through a change of training data domain: biomedical, bioBert (Lee et al., 2020); scientific publications, SciBert (Beltagy et al., 2019); knowledge graphs, Ernie (Zhang et al., 2019). Also, other variants altered the structure of the training data: moving from words to spans, SpanBert (Joshi et al., 2020); or from a linguistic input to a multimodal one, ViLBERT (Lu et al., 2019). While we use the basic model in our experiments, we consider that both lines of model development are relevant to our

further modelling plans.

Besides the fact that these models achieve state of the art performance on multiple tasks, recent research has shown that they encode a multitude of linguistic patterns to some extent (Rogers et al., 2020).

For example, Tenney et al. (2018) find that contextualized pre-trained models give big gains in performance relative to non-contextual models on morpho-syntactic tasks, such as part of speech tagging, constituent and dependency labelling. These patterns are basic components in the NLP pipeline in general, leading to a better language understanding for the model. The contextualized models show improvement also on entity labelling and relation classification, which we can consider as subcomponents of the character identification task to some extent.

Liu et al. (2019) reveals similar patterns to Tenney et al. (2018) for most of the linguistic phenomena that were probed, but they also reveal a few shortcomings: the models lose performance on tasks requiring fine-grained linguistic knowledge (e.g. conjunct identification). Also, training on in-domain data or more data in general results in a boost in performance. From a different perspective, Ettinger (2020) analyzes BERT's capacity relative to human performance. They show that BERT is competitive in detecting role reversal or retrieving noun hypernyms, but it struggles with common sense and pragmatic inference or negation, which are more complex phenomena.

Also, related to the task of character identification, BERT has a strong positive impact on coreference resolution. Joshi et al. (2019) adapts BERT for this task and they test it on 2 different datasets: paragraph-level coreference resolution (GAP Webster et al. (2018)) and document-level coreference resolution (OntoNotes Hovy et al. (2006)). BERT improves relative to previous pre-trained models, especially on distinguishing between related but different entities (for instance, President or CEO). On the other hand, it shows limitations on handling document-level context, paraphrasing or conversations. These last shortcomings are a worrying finding for our purposes, considering that we deal with sitcom dialogues.

Considering all these findings of linguistic phenomena about BERT and the analysis conducted in Chapter 3, we expect that the addition of BERT to the model proposed in Chapter 4 will bring a boost of performance on datapoints that are based on morpho-syntactic features, but it will struggle to some extent in cases that require common sense reasoning or a deeper understanding of the text.

5.3 Model

In order to combine the ability of the pre-trained language model to encode local contextual information with the global patterns developed by the entity library, we

combine these techniques in a unified architecture. Figure 5.1 highlights the main characteristics of the model: the tokens go through a pre-trained BERT, then the speaker information is added to the token representation. This representation is fed to a multi-layer perceptron (MLP). The output of this step is compared to the entity library (EntLib) in order to produce the final prediction.

The main component of the new neural network is formed by a pre-trained masked language model: the basic BERT architecture. This architecture has become a standard method for the contextualization of tokens.

The model was previously trained using two corpora: the BooksCorpus (800M words) and English Wikipedia (2,500M words) on the following two tasks:

- masked language modelling: 15% of the tokens present in the input are replaced by a **[MASK]** token and the model needs to predict the replaced token.

- next sentence prediction: the model receives two sequences as input (X_a, X_b) separated by a special token **[SEP]** and preceded by a classification token **[CLS]**. The model needs to predict if X_b is the continuation of X_a using the encoding from the classification token **[CLS]**.

While there is a multitude of BERT variants (see Section 5.2), we use the original model. BERT is a multi-layer bidirectional Transformer encoder. We use BERT-base, which is built using a stack of 12 transformer blocks (described in chapter 2) with a hidden size of 768 units and 12 attention heads. This architecture has 110 million parameters. Besides this pre-trained model, we also experiment with DistilBert (Sanh et al., 2019), which compresses the previous model by 40% while retaining 97% of its language understanding through knowledge distillation. This addition was necessary due to computation power issues, as we wanted to see how the model performance and behaviour change when it is exposed to longer sequences.

The last hidden representation of this model (emb_{tok}) has 768 dimensions and it is concatenated with a speaker embedding (emb_{sp}). The speaker embeddings are initialized through a linear mapping (equation 5.2) and their dimension size is subject to hyperparameter search with values between 100 and 400.

$$emb_{tok} = BERT(token_{id}) \tag{5.1}$$

$$emb_{sp} = speaker_{id} * W_1 \tag{5.2}$$

$$hid_1 = emb_{tok} || emb_{sp} \tag{5.3}$$

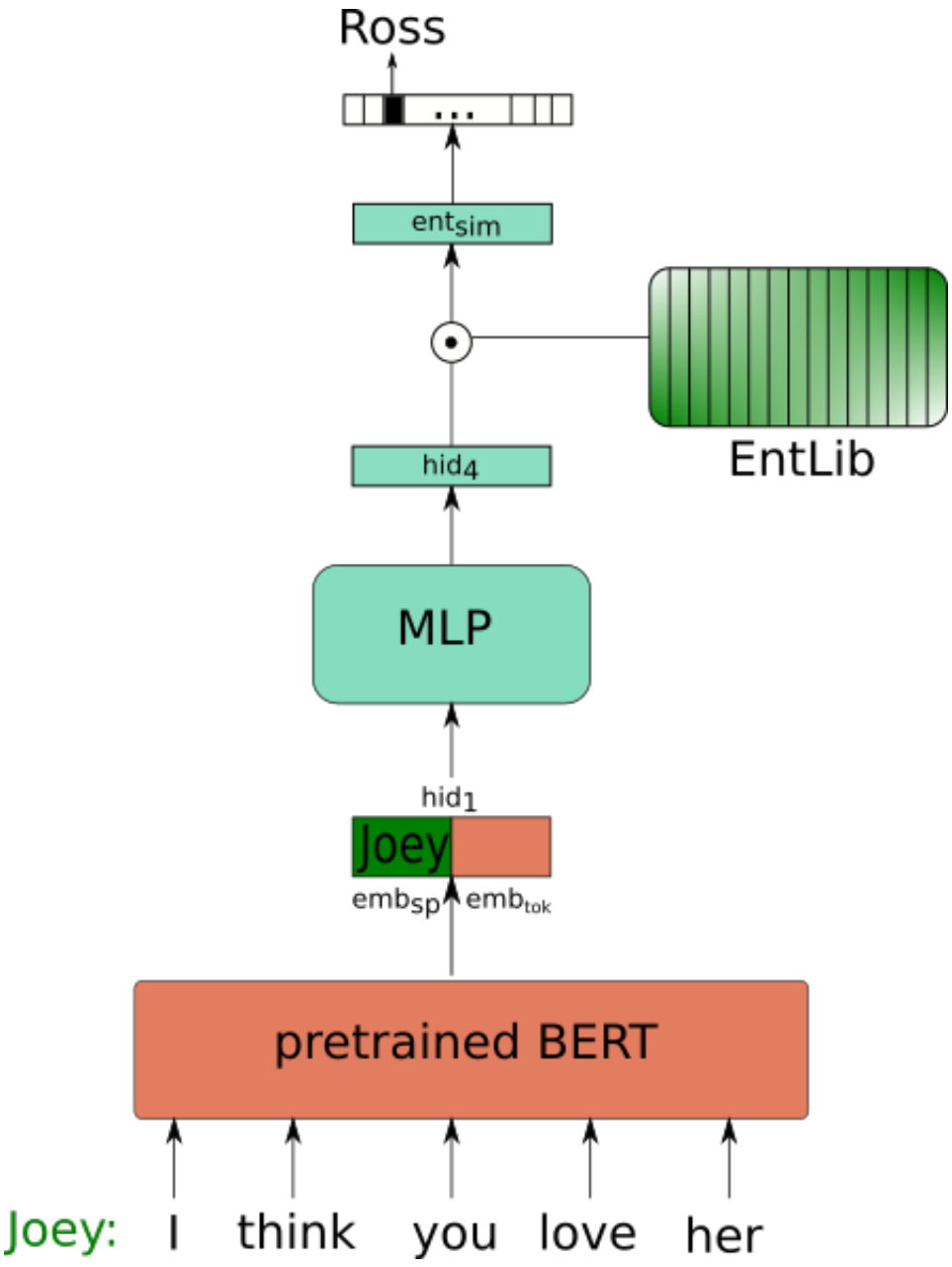

Figure 5.1: Proposed model

The concatenated representation (eq. 5.3) is fed to a 3-layer fully connected network (eqs. 5.4-5.6). This MLP is formed by nonlinear transformations. The first one reduces the dimensionality to a given hidden dimensionality (hyperparameter between 200, 300 and 400). The other layers conserve the size of the corresponding inputs.

$$hid_2 = ReLU(hid_1 * W_2) \tag{5.4}$$

$$hid_3 = ReLU(hid_2 * W_3) \tag{5.5}$$

$$hid_4 = ReLU(hid_3 * W_4) \tag{5.6}$$

Similarly to the EntLib model in chapter 4, the output of this network is compared to each entry of the entity library matrix (eq. 5.7), which was initialized with the same weights as the speaker embeddings (parameter sharing between EntLib and speaker embeddings is a part of the hyperparameter search). This comparison outputs 401 similarity scores, one for each entity. Then we apply a softmax on these scores in order to get the entity predicted by the network.

$$ent_{sim} = hid_4 \odot EntLib \tag{5.7}$$

$$out = Softmax(ent_{sim}) \tag{5.8}$$

The backpropagation is conducted on the whole network, including all the BERT components.

We conduct a set of ablation experiments in order to investigate the benefits of different components:

- **random embeddings**: the BERT component is substituted by a random embedding which linearly maps the token id into a vector of dimension between 100 and 400. Equation 5.1 is substituted by $emb_{tok} = token_{id} * W_{tok}$
- **frozen BERT**: the BERT component is not fine-tuned towards the character identification task. We continue to fine-tune all the other components of the model.
- **-EntLib**: the model does not include the entity library. Instead, the output of the MLP is directly mapped to 401 dimensions to predict the referred entity.

Currently, the speaker embedding is not contextualized because it is concatenated at the BERT output level. This limitation is a big problem in the case of second-person pronouns, especially because the entity we refer to when we use "you" is most probably an interlocutor who is the speaker of previous or future utterances. The current architecture doesn't have the ability to access this information.

In order to overcome this shortcoming, we experimented with adding a self-attention layer on top of the concatenation of the token and speaker information hid_1. The self-attention layer operates on the whole sequence given as input: it compares the hidden representation hid_1 at time step t with the hidden representation at all the other time steps. These comparisons are used to create a weighted representation.

This component can have 1, 2 or 4 attention heads and 1 or 2 layers of attention. While we expected this mechanism to help especially with second-person pronouns, this was not reflected in our hyperparameter search because our best models don't use this component. Our hypothesis for this result is that the component doesn't have a recency feature in order to focus more on the speakers surrounding the current token and instead it focuses on all the spans of BERT, which are more than 100 for our setup. It is harmful to attend equally on all the representations because in most of the cases, the referred entity with expressions like "you" is an active speaker in the vicinity of the current utterance.

5.4 Results

Because we want a comparable setup to the previous experiments, the dataset and the task are the same as the ones described in Chapter 4. The task is character identification and the training and test data for the task span the first two seasons of the sitcom Friends.

The main results of the current experiments are presented in Table 5.1. [1] The newly proposed model is comparable with the model from the previous chapter (we use the static entity library model for comparison): it is better on F1 score for all entities while it is surpassed for the other three metrics.

While the LSTM with entity library (LSTMEnt) has the best overall results, it surpasses fine-tuned BERT with the entity library (BERTEnt) only on a few input patterns. Figures 5.2 presents the F1-score for the studied models for different types of tokens: first/second/third-person pronouns, proper nouns and common nouns when we consider all the entities. Figure 5.3 looks instead at the model performance on these token types when we compress all the non-main entities in one

[1] While the prediction is over 401 entities, "all entities" in Table 5.1 are only 78 because this is the number of entities appearing in the test data.

		all (78)		main (7)	
	models	F_1	Acc	F_1	Acc
random	-EntLib	40.4	63.6	70.6	69.4
	+EntLib	43.8	64.4	71.2	70.4
BERT	frozen-EntLib	31.6	64	72.5	72.8
	frozen+EntLib	35.3	63.8	70.9	71.1
	finetuned-EntLib	38.6	62.2	68.9	69.1
	finetuned+EntLib	**51.4**	*70.5*	*76.9*	*77.6*
LSTM	+EntLib	49.6	**77.6**	84.9	**84.2**

Table 5.1: Model parameters and results on the character identification task.

class. As shown in Figures 5.2 and 5.3, LSTMEnt is better than BERTEnt on first and second-person pronouns, while the latter is better on third-person pronouns and proper nouns with similar results for common nouns. So, having a better contextualized textual representation of the input is beneficial for datapoints that require a broader linguistic understanding of the input, such as third-person pronouns. This is similar to our findings from the controlled experiments on chapter 2 where easier patterns don't require nor use contextualization, while self-attention becomes more fruitful on more complex phenomena. While each model outperforms the other in specific types of input, LSTMEnt is better generally because of the data distribution: 44.4% of the datapoints are first-person pronouns, and 27.9% are second-person pronouns.

When we focus on the models proposed in this chapter, we see a constant improvement when we add the entity library for all 3 model variations. Also, the complete model (BERTEnt) is the one reaching the best performance. This suggests that all the components are beneficial for the task.

Surprisingly, the best model has DistilBERT as its main component, which is contrary to our expectations and previous literature, which showed that bigger pre-trained models lead to better results on downstream tasks (Devlin et al., 2018). Our explanation for this result is the following: it is too hard to induce dataset or task-specific biases when you have a big amount of parameters and a very small training dataset, which is the case for the BERT model.

Also, the model initialized with the random embeddings is comparable with the model with frozen BERT embeddings. This result suggests that the encoded representations of BERT are not directly applicable to the current task and that they need to be adjusted through fine-tuning. An explanation for this pattern is that the data from dialogues, and more specifically TV sitcoms is very different in comparison to the data used for training BERT which is narrative text generally.

Furthermore, the performance of the models is different relative to the type of

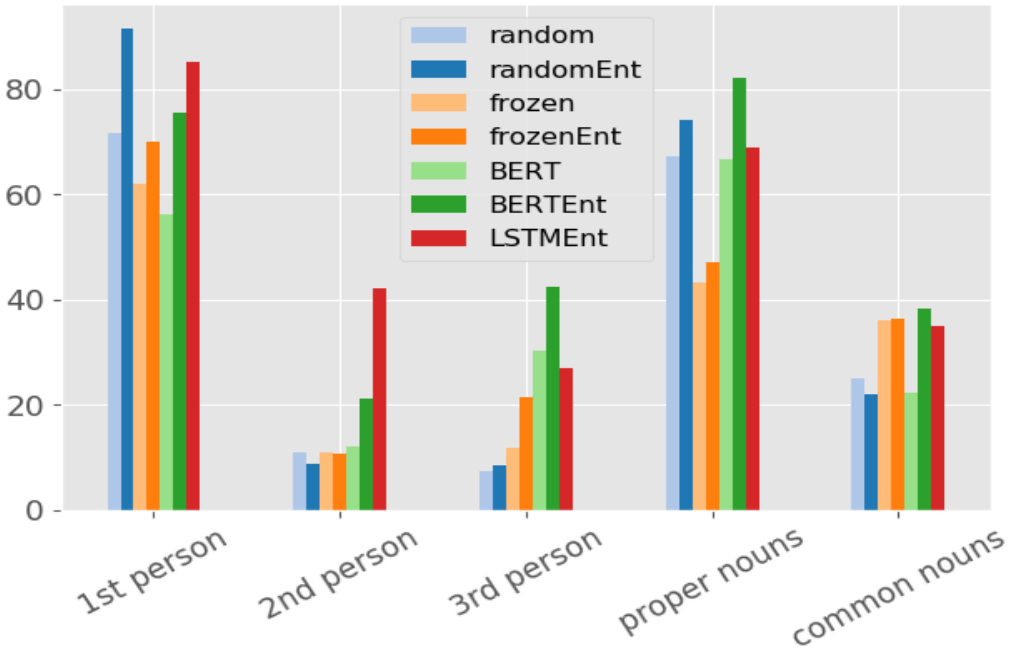

Figure 5.2: F1-score PoS analysis for new models for all entities

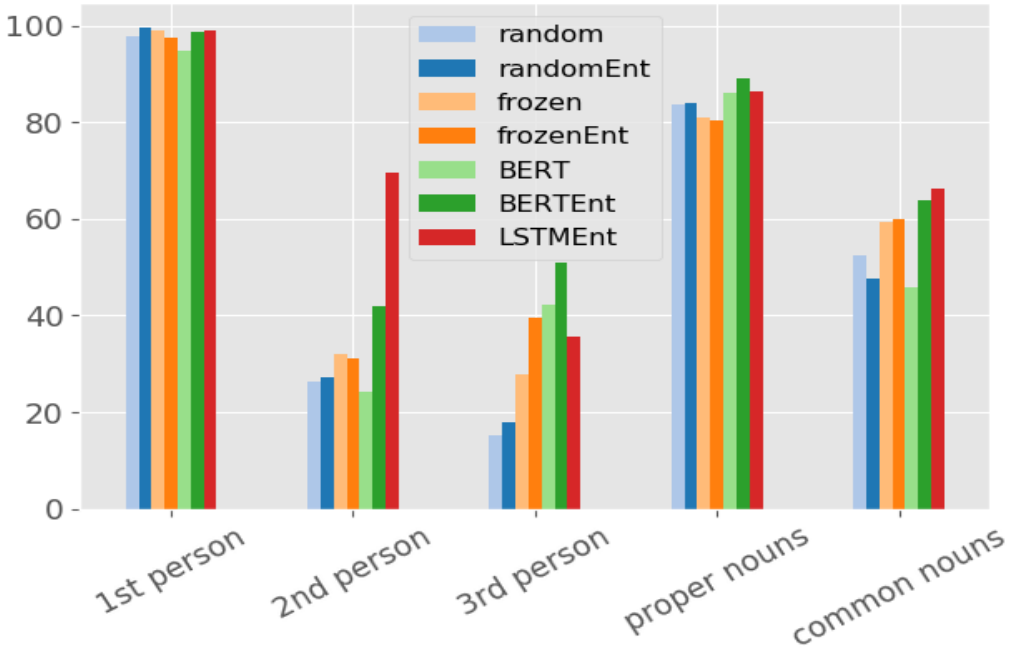

Figure 5.3: F1-score PoS analysis for new models for the main entities

the referring expression (Figure 5.2).

When we look at first-person pronouns, the best model is the model with random embeddings and entity library (randomEnt), surpassing the other variants, especially in the case of "all entities" (91.5% vs 80%). Also, we get similar results as in the previous chapter, with the entity library being a beneficial component for rare entities across all the models. For these referential expressions, the model can develop the simple mechanism that the character referred to is the same as the speaker and that the token representation is a constant that functions simply as a prompt. BERT models are constructed with the goal to contextualize the tokens, which goes against the patterns necessary for first-person pronouns, so it is harder to develop this pattern.

Proper nouns are tokens that follow a similar pattern as first-person pronouns. We don't need information from the context in order to predict which character we refer to when we say "Ross". While for first-person pronouns, we require external information about the speaker in order to predict the target character, this is not necessary for proper nouns. While the BERT-based models were not able to fully harvest the speaker information in the case of first-person pronouns, it looks that the model learns the token pattern in the case of proper nouns when we fine-tune the BERT component, surpassing the previous LSTM-based model.

Opposite to the results from the previous chapter, the results are better for third-person pronouns than for second-person pronouns for BERT-based models. This behaviour is expected considering that third-person pronouns are tokens that require extracting more information from the context in order to make a meaningful prediction, and BERT develops contextualized representations. Moreover, the third person pronouns generally follow a more direct referring expression, such as proper nouns (e.g. "**Ross** is a paleontologist. **He** is at the museum now"), information which can be harvested easily by BERT. On the other hand, the results are much worse for second-person pronouns in the case of the BERT variants. As presented in the previous section, this is a drawback of the current architecture because it can't contextualize the speaker information, which is a crucial feature for solving second-person pronoun character identification.

Lastly, the performance for common nouns is similar to the previous model. This result is unexpected because this type of referring expression requires similar reasoning as third-person pronouns, based on contextual information: We can't say to which character we refer to when we have the token "man" if we don't take into account also the words surrounding it.

Figure 5.4 presents the model accuracy for the developed models when the entities are clustered considering their frequency in the text: high frequency ($>$100 occurrences), medium frequency (between 20 and 100 occurrences) and low frequency ($<$20 occurrences). The figure shows a smoother decline for BERTEnt in comparison to LSTMEnt, which is a good pattern because it is easier to improve

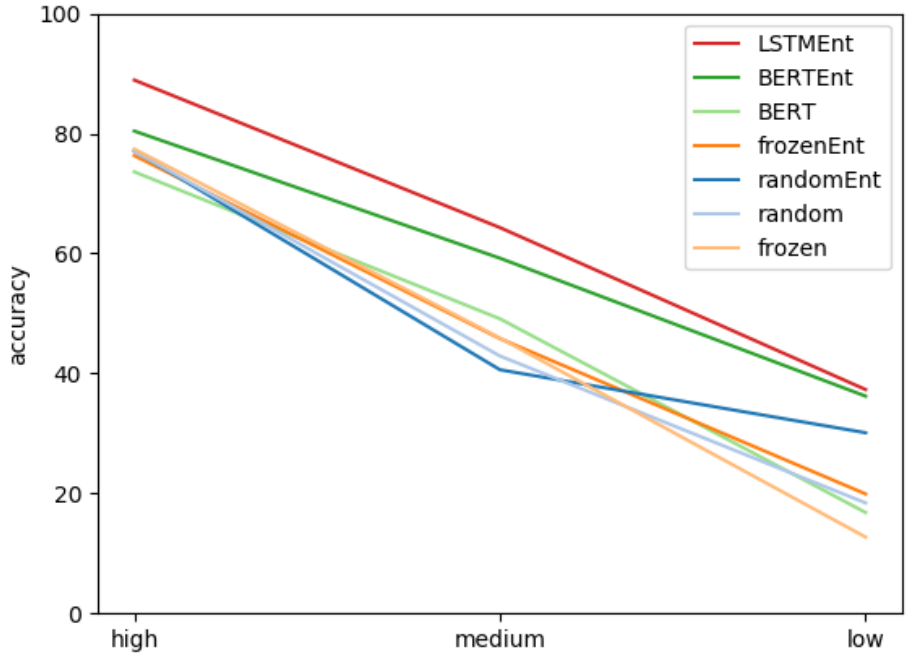

Figure 5.4: The model accuracy for entities with different levels of frequency

on more frequent entities. The two models that have different behaviour in this figure are the random model with the entity library (randomEnt) and the BERT model without the entity library (BERT). The former has a bigger drop in performance for medium frequency entities, while the latter has the opposite pattern. Our explanation for this behaviour is that medium frequency entities are more often the subject of the conversation without being present in the scene, which would lead to more contextual tokens for these entities.

To bring more insights on the model prediction, we also look at some patterns for BERTEnt in Tables 5.2 and 5.3. Table 5.2 presents the type of the predicted entity (main or not main) relative to the type of the target entity when the model doesn't have a correct answer. Table 5.3 shows the gender of the predicted entities when the model makes a mistake relative to the gender of the target entity.

First, Table 5.2 shows that the model tends to predict main entities even when it is not correct, which is expected considering that 70% of the training data refers to these 6 main entities. On the other hand, we expected that the models would

[1]we didn't include the datapoints that have as input the following third-person pronouns: it, itself

prediction ⇒ target ⇓	main ent	other ent
main ent	289	45
other ent	210	172

Table 5.2: BERTEnt mistakes relative to main/not main entities

prediction ⇒ target ⇓	masculine	feminine
masculine	155	81
feminine	159	68

Table 5.3: BERTEnt mistakes for third person pronouns[2]

develop gender associations for the characters because this information is given by third-person pronouns directly: for example, it is more probable to predict "Chandler" or "Joey" than "Rachel" when the model doesn't predict correctly "Ross". Contrary to our expectations, the results from Table 5.3 reveal that the model will predict more often male characters than female characters, even in cases where you have referring expressions like "she", "herself" or "her". So, the pattern follows more the training data frequency [3] than the character gender association.

5.5 Analysis

5.5.1 Layer structure

Similar to the analysis from the previous chapter, we plot t-SNE projections (Van der Maaten and Hinton, 2008) of the BERT output (em_{tok}) and the output of the MLP (hid_4) in order to get more insights into the flow of information through the network. For both Figure 5.5 and 5.6, we present the projections for the BERT output on the left side of the figure, and we plot the projections for the MLP output on the right side of the figure.

First, we look at the layer structure when the input is a first-person pronoun (Figure 5.5). The output of BERT doesn't manifest any structure, which is expected considering that BERT is not exposed to the speaker information and the context for first-person pronouns is not very meaningful in general. After the information passes through the MLP, we see very clean clusters associated with the main entities while all the other characters form another cluster. This pattern suggests that the MLP assures full integration of the speaker information into the representation.

Second, we look at the layer structure for all the tokens that are not first-person pronouns (Figure 5.6). The BERT representations generate slight clusters this time. An explanation for this pattern is that the input either carries information

[3]there are 947 masculine utterances in the training data in comparison with 728 feminine utterances

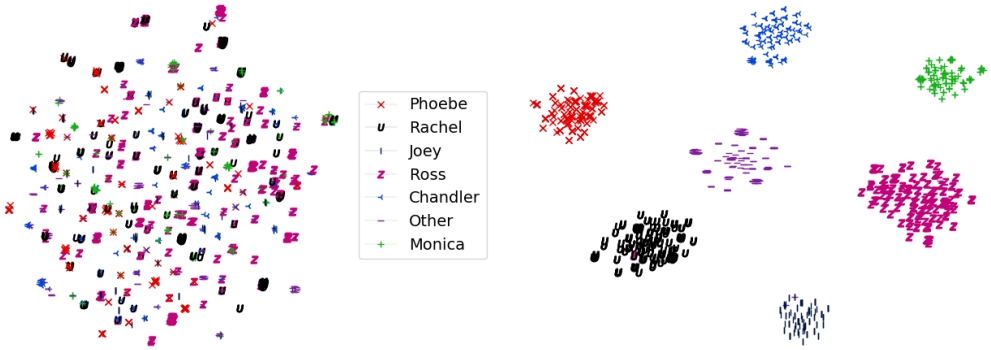

Figure 5.5: BERTENT, 2D TSNE projections of the activations for first-person mentions in the test set, colored by the entity referred to. BERT output emb_{tok} left graph; MLP output hid_4 shown in the right graph. Best viewed in color.

characteristics to an entity in the case of proper nouns and common nouns to some extent or that the context information is harvested. In this case, we see that the structure doesn't change very much between BERT representations and MLP representations, which suggests that the main role of the MLP is to integrate the speaker information when it is necessary.

5.5.2 Entity representations

We conduct the same analysis as in Chapter 4 where we have developed sentences for probing entity information, such as "This **person** is a singer-songwriter", which requires the prediction for "person" to be "Phoebe Buffay".

Additionally, we conduct experiments where we substitute "This person" with third-person pronouns.

We do this change in 2 different ways:

1. we substitute the "this person" component with the third-person pronoun associated with the character ("he" for men and "she" for women). For example, the previous example becomes "**she** is a singer-songwriter" and we have "**he** is a manager" for Chandler Bing.

2. we substitute the "this person" component with the incorrect third-person pronoun. We use "he" for female characters and "she" for male characters. The previous examples become "**he** is a singer-songerwriter" and "**she** is a manager".

The purpose of these additional setups is two-folded: first, the model is more familiar with third-person pronouns than the expression "this person" because

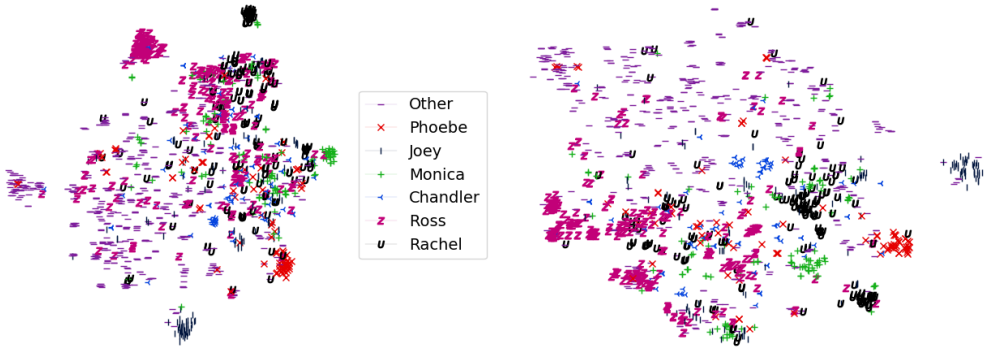

Figure 5.6: BERTENT, 2D TSNE projections of the activations for mentions in the test set (excluding first-person mentions), colored by the entity referred to. BERT output emb_{tok} left graph; MLP output hid_4 shown in the right graph. Best viewed in color.

	This person	Correct pronoun	Wrong pronoun
Accuracy	2.9	3.8	1.9

Table 5.4: Entity representation probing for BERTEnt

they are much more frequent in the training data. Second, it also gives us insights regarding gender encoding in the network: we hypothesise that a model that encodes gender information will exhibit a better performance when the correct gender information is added to the probing sentences while adding the wrong gender pronoun should be detrimental.

Even though BERTEnt has slightly better accuracy than LSTMEnt (2.9% v.s. 2.2%), the new model still has a very poor performance when we look at the general results (Table 5.4). While it is not the desired result, it is also not a very surprising behaviour. Even for humans, it is hard to remember that we refer to "**Terry**" when we say "This person is a coffee shop owner" if we saw Terry only once when we watch the TV show. On the other hand, the model is sensitive to some extent to the gender information added by third-person pronouns. We have almost one percent improvement when we substitute the previous referential phrase with the correct third-person pronoun, going from 2.9% to 3.8%. When we use the wrong pronoun as the referential expression, the accuracy drops to 1.9%, indicating that this information is detrimental to the predictions.

5.6 Discussion

The hypothesis that we had when designing BERTEnt was that the new architecture would surpass in performance the previous model by having the same performance as LSTMEnt on extra-linguistic patterns while improving contextualization and building more meaningful entity representations.

Our proposed model is solving, to some extent, the referring expressions that require contextualization, improving on third-person pronouns (however, it does not improve on common nouns). The model is also developing the token-based patterns we identified in Chapter 4, such as those that allow for the solving of proper nouns. In the case of extra-linguistic patterns related to first-person pronouns, the model doesn't find the complete balance exhibited by models that don't contextualize (the random model), but it is very competitive. Most importantly, the model is not able to contextualize the speaker information because this information is not a component of BERT; this leads to a big drop in performance for second-person pronouns.

Even though the proposed model doesn't surpass LSTMEnt, both the entity library and fine-tuning BERT prove to be beneficial when we compare the proposed model to the ablated ones. First, fine-tuning BERT finds a balance between the behaviour of the model with random embeddings and the behaviour of the model with frozen representations. Second, similar to the findings from Chapter 4, the entity library proves to be a valuable additional module. It boosts the performance for rare entities and improves the results across most of the types of referring expressions. Furthermore, future work should experiment with the dynamic entity library in this setup. The contextualized embeddings produced by the BERT component could enhance the ability to capture episodic information of the dynamic part of the additional module.

Finally, the model doesn't develop representations that encode basic entity information, such as attributes or relations when we refer to entities. While we expected the development of general entity representations, this is the hardest feature to learn. Also, this feature might be impossible to learn with the amount of data available; or in the context of this task (though we hypothesize that the relatively low amount of data, rather than the task, is responsible for the results).

While the model develops in a good direction, there are still clear ways to improve it: most prominently, a better way to incorporate extra-linguistic information such as speaker information or visual information. While the need to add this kind of information is clear, it is not so clear how to do it; a large amount of training data is required to pre-train a model like BERT. Another possible model development is to expose the model to the domain language first by either fine-tuning the BERT component on the task of masked language modelling on TV dialogues or designing a multi-task fine-tuning with both character identification

and language modelling as objectives. We think that more familiarity with the TV show language or more training data would have a big impact on improving the performance of the proposed model.

Chapter 6

CONCLUSION

The goals of this dissertation are to have a better grasp of the entity information encoded in current computational models and develop a performant computational model of reference to entities.

We developed a set of datasets and procedures that can be used to detect the degree of referential patterns encoded in computational models, both in a controlled and in a more scaled and natural setup.

In the second part of the thesis, we proposed a set of computational models with the ability to refer to entities and to build entity representations, and we incorporated them into current neural network architectures. Through various analysis studies, we showed both the benefits and the weaknesses of the proposed models.

The rest of the chapter focuses on summarising and discussing the main findings and contributions of the thesis (Section 6.1), and on suggestions for future research (Section 6.2) concerning the two research goals stated in the Introduction.

6.1 Main findings

Goal 1: To analyze the entity encoding capabilities of neural networks. First, we investigated the capabilities of standard architectures to encode different patterns relevant to entity-related tasks when these models are trained from scratch in a controlled environment. We found that feature detection and extraction from sequences can be solved with a simple attention mechanism. On the other hand, the structure of current standard architectures encountered problems when they had to capture more complex patterns such as contextualization or order tracking. Furthermore, the models were negatively affected when the sequence increased in length. Contextualization, order tracking, and longer sequences are all relevant properties of entity-related tasks.

We next turned to the analysis of the referential capabilities of language models. This is because we hypothesized that transfer learning could alleviate some of the shortcomings presented by the standard architectures when they are trained from scratch. This strategy implies training a part of a model on a different task first, where we have access to more data. The most common pre-training task for transfer learning is the task of neural language modelling.

When we analyzed the types of entity-related patterns present in neural language models, we determined that these models encode local contextual patterns very well. One linear-layer transformation learns to do anaphora resolution, with a strong detection of gender and number agreement constraints. This was as expected given previous work on language models. What was more surprising was the fact that, even though it is very hard to capture a more global notion of entity, neural language models develop entity-specific information to some extent. First, we discovered that, while the models are confused by mentions in the context that are not antecedents, they are still much better than baselines. Second, the models could predict when two pronouns are referring to the same entity, though only after we controlled both for distance and for pronoun form. In summary, we found that pre-trained language models provide good representations of the contextual linguistic input concerning referential information, and that they encode entity-specific information to some extent. Considering all these factors, we concluded that we could use transfer learning from language modelling for models developed for entity-related tasks.

Goal 2: To develop computational models with the capacity to encode both linguistic and extra-linguistic referential information. In the first part, we developed two types of structures that should suit the encoding of entities: a static one (EntLib) and a dynamic one (EntNet). The static module is a matrix where each vector is associated with an entity, and these vectors are updated only through backpropagation. In the case of the dynamic module, each entity is represented by a static "key" vector, which is similar to the EntLib vectors, and a dynamic "value" vector which is updated at every step with new information. We hypothesized that the static library would develop a global semantic representation for each entity which would contain the main characteristics of each entity like gender, job or relations to other characters; and that the dynamic structure would encode the semantic global information in its key, while it would capture the situational/contextual information about an entity in the "value" vector associated with a specific entity.

Our findings were surprising, not following our initial hypothesis. First, there is no difference in results or behaviour between the model enhanced with the static structure in comparison with the dynamic one, suggesting that the "value" com-

ponent was not updating as intended. Furthermore, the vectors associated with the entities didn't encode crucial characteristics such as gender, job or relationships. However, the addition of the entity-specialized structures did help with the handling of extra-linguistic information, such as a good integration of the speaker information and the association of an entity label with the correct name.

We considered that the weak representations in the first round of model development might be caused by the fact that in the character identification dataset that we used, there was not enough training data to learn, from scratch, both morpho-syntactic and global entity patterns. Following the analysis of pre-trained language models from Chapter 3, we hypothesized that using transfer learning from pre-trained contextualized language models could overcome the shortcomings of the previously developed models. Pre-trained contextualized language models (e.g. BERT) provide strong contextual representations and general lexical patterns, while specialized structures like EntLib and EntNet provide a good mechanism to capture extra-linguistic information.

Our second hypothesis was that the combination of these two mechanisms (transfer learning and specialized modules) would bring a boost in performance in the character identification task. Furthermore, we considered that taking the burden away from the main network and providing richer lexical representation would lead to better entity representations. Our results didn't fully match our expectations. As expected, the model improved on references that require more contextual information, such as third-person pronouns, in comparison with the models trained from scratch. However, the main difficulty we encountered is the fact that pre-trained models come with a predefined structure that can't be changed easily. Generally, they are trained on running text, with the input always being formed by words, without knowledge of speakers or other extra-linguistic information. This limitation affected our model development because the speaker information was not contextualized, which lead to a loss of interlocutor awareness. This model flaw generated a decrease in performance in second-person pronoun referents which counterbalanced the improvement brought about by third-person pronoun referents. Regarding the entity representations developed in the dedicated slots, they improve relative to the previous experiments, but they are still far from being satisfactory.

6.2 Limitations and future work

We discuss future research directions related to the two goals of the thesis. In the discussion, we use the same structure as in the previous section for the sake of clarity.

Goal 1: To analyze the entity encoding capabilities of neural networks. The first line of experiments built a controlled environment where we could isolate and study different linguistic phenomena without the interference of unwanted factors. The tested models struggled to solve the tasks we defined. Future work can expand on our framework in two different directions. First, they can study the effect of scaling up the tested models such that they are closer to the size of state-of-the-art models, to see whether this change leads to a big improvement in performance in the currently defined tasks. A second direction is to test whether changes to the architectures bring models closer to solving the tasks, thus capturing the studied phenomena. Also, the flexible construction of the controlled environment offers the possibility of extending the current datasets to more features, different sequence lengths, a different number of training examples, and new task types covering different linguistic phenomena.

In the second round of analysis studies, we developed a pipeline to analyse the referential information in pre-trained contextualized language models. Pre-trained language models keep changing, and further research should apply our analysis pipeline to newer pre-trained models. This additional analysis could reveal which changes in training data or model structure lead to different encodings relative to referential patterns, and these insights can be translated into design decisions for newer architectures. Future work can also extend the analysis from narrative texts to dialogue. This change would give a better understanding of the impact of speaker information, as well as how the frequency of different referring expressions impacts the analysis of referential information: narrative texts are dominated by third-person pronouns and common nouns, while there are more first- and second-person pronouns in dialogue.

Goal 2: To develop computational models with the capacity to encode both linguistic and extra-linguistic referential information. We developed two entity-focused modules that can be connected to different computational models. This addition brings an improvement on rare entities and in capturing extra-linguistic patterns. As shown through our experiment, the proposed architectures have strong general results on the task of character identification. In future work, it would be advisable to test the proposed architectures on different entity-related tasks that can benefit from the specialized structure, such as question answering and coreference resolution.

In the case of the developed specialized entity structures, an aspect left for future research is a better understanding of the causes for the dynamic model to converge to a similar behaviour to the static model. This can be achieved by further analysis, testing the architecture on different datasets, and/or connecting the module to a different linguistic context encoder.

An important factor for the results in deep learning is the amount and the structure of the data used for training. While we consider the task of character identification a suitable one to test reference to entities and to encourage computational models to build entity representations, an increase of annotated data and the inclusion of visual information would be necessary to make stronger claims about model development.

Regarding the development of the challenge set used for probing the entity information in the models trained on character identification, the models reach low performance on this task; however, we argue that it is a difficult task even for humans. Therefore, future work should collect human annotations for this dataset to obtain a more complete view of the difficulty of the challenge set and to allow a more thorough analysis. Prior to the collection of human annotations, it would be important to determine which are the most salient characteristics of each entity; these properties can be expected to be the most important for the entity library to capture, in order to distinguish between characters. Note, however, that the data collection is challenging because it requires prior knowledge of the sitcom.

In the case of the model that combines contextualized language models with the entity structures, the BERT component is not as powerful as we expected. Our conjecture for this performance is the fact that the data used for training BERT is very different in comparison with the new input. The first step after the experiments presented in the thesis should be to first fine-tune the BERT component using all the series seasons and then move to the task of character identification. This change requires only access to the raw utterances from the TV show without further annotation. While this is an immediate solution, it will probably still not solve the lack of awareness of interlocutors in the case of deployed models. In order to address this issue, a possible approach for future research would be to develop a model similar to ViLBERT (Lu et al., 2019), which extends the classic BERT architecture to a multi-modal two-stream model. With access to a large amount of dialogue data with speaker annotation, one can train a two-stream model with two parallel BERT-style models operating over speaker information and text segments. This change in BERT structure should lead to a better blending of speaker information with linguistic information, which is necessary for a better grasp of the entities participating in a conversation.

Even though the dynamic entity library was not successful in the experiments conducted in Chapter 4, its structure could be more suitable if used in conjunction with a pre-trained language model like BERT, which provides a good contextualized representation of the input. The dynamic component of the entity library could ensure that the contextual information outputted by BERT is grounded in the space of entities by detecting the referred entity and updating its episodic representation. Further experiments with this approach might lead to a behaviour closer to our expectations regarding this architecture.

Bibliography

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Agrawal, A., Batra, D., and Parikh, D. (2016). Analyzing the behavior of visual question answering models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1955–1960, Austin, Texas. Association for Computational Linguistics.

Alishahi, A., Belinkov, Y., Chrupała, G., Hupkes, D., Pinter, Y., and Sajjad, H., editors (2020). *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Online. Association for Computational Linguistics.

Alishahi, A., Chrupała, G., and Linzen, T. (2019). Analyzing and interpreting neural networks for nlp: A report on the first blackboxnlp workshop. *Natural Language Engineering*, 25(4):543–557.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *Iclr 2015*, pages 1–15.

Bansal, T., Neelakantan, A., and McCallum, A. (2017). RelNet: End-to-End Modeling of Entities & Relations. *CoRR*, abs/1706.07179.

Belinkov, Y. and Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. 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), pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

Bernardi, R., Boleda, G., Fernández, R., and Paperno, D. (2015). Distributional semantics in use. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 95–101, Lisbon, Portugal. Association for Computational Linguistics.

Broscheit, S. (2019). Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 677–685.

Bunescu, R. and Paşca, M. (2006). Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proc. of EACL*.

Chen, H., Wei, B., Liu, Y., Li, Y., Yu, J., and Zhu, W. (2018). Bilinear Joint Learning of Word and Entity Embeddings for Entity Linking. *Neurocomputing*, 294:12–18.

Chen, H. Y., Zhou, E., and Choi, J. D. (2017). Robust Coreference Resolution and Entity Linking on Dialogues: Character Identification on TV Show Transcripts. In *Proc. of CoNLL 2017*.

Cheng, P. and Erk, K. (2019). Attending to entities for better text understanding. *arXiv preprint arXiv:1911.04361*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Choi, J. D. and Chen, H. Y. (2018). SemEval 2018 Task 4: Character Identification on Multiparty Dialogues. In *Proc. of SemEval*.

Chrupała, G. and Alishahi, A. (2019). Correlating neural and symbolic representations of language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2952–2962, Florence, Italy. Association for Computational Linguistics.

Clark, E., Ji, Y., and Smith, N. A. (2018). Neural Text Generation in Stories Using Entity Representations as Context. In *Proc. of NAACL*.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019). What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Clark, K. and Manning, C. D. (2016). Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *Proc. of ACL*.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018a). 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 (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018b). What You Can Cram into a Single $&!#* Vector: Probing Sentence Embeddings for Linguistic Properties. In *Proc. of ACL*.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2018c). Word Translation Without Parallel Data. In *ICLR 2018*.

Craswell, N. (2009). Mean Reciprocal Rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019a). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., and Salakhutdinov, R. (2019b). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2017). Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Ettinger, A. (2020). What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Francis-Landau, M., Durrett, G., and Klein, D. (2016). Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks. In *Proc. of NAACL:HLT*.

Futrell, R., Wilcox, E., Morita, T., and Levy, R. (2018). RNNs as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

Giulianelli, M., Harding, J., Mohnert, F., Hupkes, D., and Zuidema, W. (2018a). Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Giulianelli, M., Harding, J., Mohnert, F., Hupkes, D., and Zuidema, W. (2018b). Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248.

Glockner, M., Shwartz, V., and Goldberg, Y. (2018). Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th*

Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309.

Goldberg, Y. (2019). Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Graves, A., Fernández, S., and Schmidhuber, J. (2005). Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *Proc. of ICANN*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Gulordava, K., Aina, L., and Boleda, G. (2018a). How to Represent a Word and Predict it, too: Improving Tied Architectures for Language Modelling. In *Proc. of EMNLP*.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018b). 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, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018c). Colorless Green Recurrent Networks Dream Hierarchically. In *Proc. of NAACL*.

Haghighi, A. and Klein, D. (2010). Coreference Resolution in a Modular, Entity-Centered Model. In *Proc. of NAACL*.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proc. of ICCV*.

Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. (2016). Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. (2017). Tracking the World State with Recurrent Entity Networks. In *Proc. of ICLR*.

Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. (2019). Tracking the world state with recurrent entity networks. In *5th International Conference on Learning Representations, ICLR 2017*.

Hewitt, J. and Liang, P. (2019). Designing and interpreting probes with control tasks. 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)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2020). Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Hupkes, D., Veldhoen, S., and Zuidema, W. (2018a). Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Hupkes, D., Veldhoen, S., and Zuidema, W. (2018b). Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

İrsoy, O. and Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, Doha, Qatar. Association for Computational Linguistics.

Ji, Y., Tan, C., Martschat, S., Choi, Y., and Smith, N. A. (2017). Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Joshi, M., Levy, O., Zettlemoyer, L., and Weld, D. (2019). BERT for coreference resolution: Baselines and analysis. 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)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Joulin, A. and Mikolov, T. (2015). Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. *arXiv*, pages 1–10.

Jumelet, J., Zuidema, W., and Hupkes, D. (2019). Analysing neural language models: Contextual decomposition reveals default reasoning in number and gender assignment. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 1–11.

Kádár, A., Chrupała, G., and Alishahi, A. (2017). Representation of Linguistic Form and Function in Recurrent Neural Networks. *Computational Linguistics*, 43(4):761–780.

Kamp, H. and Reyle, U. (2013). *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, volume 42. Springer Science & Business Media.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding groups in data: an introduction to cluster analysis*. John Wiley, New York.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational Similarity Analysis – Connecting the Branches of Systems Neuroscience. *Frontiers in Systems Neuroscience*, 2:4.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuncoro, A., Dyer, C., Hale, J., Yogatama, D., Clark, S., and Blunsom, P. (2018). LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Lakretz, Y., Kruszewski, G., Desbordes, T., Hupkes, D., Dehaene, S., and Baroni, M. (2019). The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Lehmann, S., Oepen, S., Regnier-Prost, S., Netter, K., Lux, V., Klein, J., Falkedal, K., Fouvry, F., Estival, D., Dauphin, E., Compagnion, H., Baur, J., Balkan, L., and Arnold, D. (1996). TSNLP - test suites for natural language processing. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., and Luís, T. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Linzen, T., Chrupała, G., Belinkov, Y., and Hupkes, D., editors (2019). *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy. Association for Computational Linguistics.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019). Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Lu, D., Whitehead, S., Huang, L., Ji, H., and Chang, S.-F. (2018). Entity-aware image caption generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4013–4023, Brussels, Belgium. Association for Computational Linguistics.

Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). Vilbert: pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 13–23.

McCoy, T., Pavlick, E., and Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mihalcea, R. and Csomai, A. (2007). Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proc. of CIKM*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013a). Distributed Representations of Words and Phrases and Their Compositionality. In *Proc. of NIPS*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. In *Proc. of NAACL*.

Mitkov, R. (2002). *Anaphora Resolution*. Studies in Language and Linguistics. Longman.

Müller, M., Rios, A., Voita, E., and Sennrich, R. (2018). A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 61–72, Brussels, Belgium. Association for Computational Linguistics.

Nematzadeh, A., Burns, K., Grant, E., Gopnik, A., and Griffiths, T. (2018). Evaluating Theory of Mind in Question Answering. In *Proc. of EMNLP*.

Noreen, E. (1989). *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley.

Paperno, D. (2014). Typology of adjectives benchmark for compositional distributional models. In *LREC*.

Park, C., Song, H., and Lee, C. (2018). KNU CI System at SemEval-2018 Task4: Character Identification by Solving Sequence-Labeling Problem. In *Proc. of SemEval*.

Parvez, M. R., Chakraborty, S., Ray, B., and Chang, K.-W. (2018). Building language models for text with named entities. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2373–2383.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037.

Peters, M., Neumann, M., Zettlemoyer, L., and Yih, W.-t. (2018a). Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations.

Poesio, M., Grishina, Y., Kolhatkar, V., Moosavi, N., Roesiger, I., Roussel, A., Simonjetz, F., Uma, A., Uryupina, O., Yu, J., and Zinsmeister, H. (2018). Anaphora resolution with the ARRAU corpus. In *Proceedings of the First Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 11–22, New Orleans, Louisiana. Association for Computational Linguistics.

Poesio, M., Stuckardt, R., and Versley, Y. E. (2016). *Anaphora resolution: Algorithms, resources, and applications*. Springer.

Poliak, A., Haldar, A., Rudinger, R., Hu, J. E., Pavlick, E., White, A. S., and Van Durme, B. (2018). Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 337–340, Brussels, Belgium. Association for Computational Linguistics.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., and Xue, N. (2011a). CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proc. of CoNLL*.

Pradhan, S., Ramshaw, L. A., Marcus, M. P., Palmer, M., Weischedel, R. M., and Xue, N. (2011b). Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011*, pages 1–27.

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., and Kim, B. (2019). Visualizing and measuring the geometry of BERT. In *Advances in Neural Information Processing Systems*, pages 8592–8600.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2021). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Ruder, S. (2019). *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway.

Sanchez, I., Mitchell, J., and Riedel, S. (2018). Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1975–1985, New Orleans, Louisiana. Association for Computational Linguistics.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sennrich, R. (2017). How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Shi, X., Padhi, I., and Knight, K. (2016). Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Sukthanker, R., Poria, S., Cambria, E., and Thirunavukarasu, R. (2018). Anaphora and coreference resolution: A review. *arXiv preprint arXiv:1805.11824*.

Tenney, I., Das, D., and Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Tenney, I., Das, D., and Pavlick, E. (2019b). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Van Durme, B., Bowman, S. R., Das, D., et al. (2018). What do you learn from

context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

van Schijndel, M., Mueller, A., and Linzen, T. (2019). Quantity doesn't buy quality syntax with neural language models. 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)*, pages 5835–5841.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Veldhoen, S., Hupkes, D., and Zuidema, W. H. (2016). Diagnostic classifiers revealing how neural networks process hierarchical structure. In *CoCo@ NIPS*.

Voita, E. and Titov, I. (2020). Information-theoretic probing with minimum description length.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Webster, K., Recasens, M., Axelrod, V., and Baldridge, J. (2018). Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., et al. (2013). Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

White, J. C. and Cotterell, R. (2021). Examining the inductive bias of neural language models with artificial languages. *arXiv preprint arXiv:2106.01044*.

Wilcox, E., Levy, R., Morita, T., and Futrell, R. (2018). What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221, Brussels, Belgium. Association for Computational Linguistics.

Wiseman, S., Rush, A. M., and Shieber, S. M. (2016). Learning Global Features for Coreference Resolution. In *Proc. of NAACL:HLT*.

Wolf, T. (2019). Some additional experiments extending the tech report "assessing BERT's syntactic abilities" by Yoav Goldberg. Technical report.

Yang, Z., Blunsom, P., Dyer, C., and Ling, W. (2016). Reference-aware language models. *arXiv preprint arXiv:1611.01628*.

Yang, Z., Blunsom, P., Dyer, C., and Ling, W. (2017). Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. (2019). ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Zhou, H., Zhang, Y., Huang, S., and Chen, J. (2015). A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China. Association for Computational Linguistics.

Zipf, G. K. (1949). *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge.