Combining Contextualized and Non-Contextualized Embeddings for Domain Adaptation and Beyond

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität München

vorgelegt von
Nina Mareike Pörner

München, den 10. Dezember 2020

Erstgutachter: Prof. Dr. Hinrich Schütze
Zweitgutachter: Prof. Dr. Anders Søgaard
Drittgutachter: Lecturer Dr. Tim Rocktäschel

Tag der Einreichung: 10. Dezember 2020
Tag der mündlichen Prüfung: 04. März 2021

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig ohne unerlaubte Beihilfe angefertigt ist.

München, den 10. Dezember 2020

Nina Mareike Pörner

Summary

Many of the recent advances in natural language processing have been spearheaded by the transfer learning paradigm, where models are pretrained on source tasks, in order to improve performance on target tasks. Pretrained models can be divided into those that produce **non-contextualized embeddings**, and those that produce **contextualized embeddings**. The former are vector representations that depend only on the word type of a given word, while the latter depend on its intra-sentential context.

Contextualized embeddings are usually produced by deep neural networks with many parameters, which can be expensive to pretrain. As a result, practitioners often re-use model checkpoints from bigger research labs and companies. This approach is not optimal in cases where there is a mismatch between the source and target domains. By contrast, non-contextualized embeddings are shallow and cheaper to pretrain, but they tend to be less successful than their contextualized counterparts.

In this thesis, we explore ways of getting the best of both worlds, by **combining contextualized and non-contextualized embeddings**. Generally speaking, our aim is to leverage the expressiveness of contextualized embeddings (e.g., BERT), while enhancing them with inexpensive non-contextualized embeddings (e.g., Word2Vec). Our contributions can be divided into two avenues:

In Chapters 2 and 3, **we align non-contextualized embeddings with the input layer of BERT**. In Chapter 2, we use this method to inject domain-specific biomedical Word2Vec embeddings into the general-domain BERT model. When evaluated on biomedical named entity recognition and question answering, the resulting model consistently improves over BERT. In Chapter 3, we use the same method to inject entity embeddings into BERT. We show that the resulting model compares favorably against variants of BERT that were explicitly pretrained to process entity embeddings.

In Chapters 4 and 5, **we align different contextualized and non-contextualized sentence embeddings** via generalized canonical correlation analysis and other sentence meta-embedding methods. In Chapter 4, we evaluate our approach on unsupervised duplicate question detection in low-resource, highly domain-specific community question answering forums. In Chapter 5, we use sentence meta-embeddings to set a new state of the art on the unsupervised semantic textual similarity benchmark.

Zusammenfassung

Viele Fortschritte der letzten Jahre in der maschinellen Sprachverarbeitung beruhen auf der Technik des Transfer-Lernens. Beim Transfer-Lernen werden Modelle auf Quell-Aufgaben vortrainiert, um die Genauigkeit auf Ziel-Aufgaben zu verbessern. Vortrainierte Modelle können unterteilt werden in solche, die **nicht-kontextualisierte Repräsentationen** produzieren, und solche, die **kontextualisierte Repräsentationen** produzieren. Erstere sind Vektoren, die nur von der Identität einzelner Worte abhängen. Letztere hängen vom Kontext innerhalb des Satzes ab.

Kontextualisierte Repräsentationen werden normalerweise von tiefen neuronalen Netzen mit vielen Parametern produziert. Das Vortrainieren dieser Modelle ist teuer. Daher werden häufig Modelle wiederverwendet, die von größeren Forschungsinstituten und Firmen vortrainiert wurden. Dieser Ansatz ist allerdings nicht optimal in Situationen, wo Quell- und Ziel-Domäne nicht zusammenpassen. Auf der anderen Seite sind nicht-kontextualisierte Repräsentationen weniger tief und weniger teuer, aber in der Regel auch weniger erfolgreich.

In dieser Dissertation holen wir das Beste aus beiden Welten heraus, indem wir **kontextualisierte mit nicht-kontextualisierten Repräsentationen kombinieren**. Unser Ziel ist, die Ausdrucksstärke von kontextualisierten Repräsentationen (z.B. BERT) zu nutzen, und mithilfe von günstigen, nicht-kontextualisierten Repräsentationen (z.B. Word2-Vec) zu verbessern. Unsere Beiträge können in zwei Richtungen eingeteilt werden:

In Kapiteln 2 und 3 **alignieren wir nicht-kontextualisierte Repräsentationen mit der Eingabe-Schicht von BERT**. In Kapitel 2 nutzen wir die Methode, um domänenspezifische Word2Vec-Repräsentationen in das BERT-Modell zu injizieren. Wir evaluieren das resultierende Modell auf biomedizinischer Entitätenerkennung und der Beantwortung von Fragen, mit Verbesserungen gegenüber BERT. In Kapitel 3 nutzen wir die Methode, um Entitäten-Repräsentationen in das BERT-Modell zu injizieren. Wir zeigen, dass das resultierende Modell kompetitiv ist gegenüber BERT-Varianten, die explizit dafür vortrainiert wurden, Entitäten-Repräsentationen zu verarbeiten.

In Kapiteln 4 und 5 **alignieren wir kontextualisierte und nicht-kontextualisierte Satz-Repräsentationen** mithilfe der generalisierten kanonischen Korrelationsanalyse und anderen Methoden. In Kapitel 4 evaluieren wir unseren Ansatz auf der unüberwachten Erkennung von Duplikat-Fragen in datenarmen, sehr domänen-spezifischen Frage-und-Antwort-Foren. In Kapitel 5 setzen wir einen neuen Bestwert auf der unüberwachten semantischen Text-Ähnlichkeits-Benchmark.

Danksagung

Meinem Betreuer, Prof. Dr. Hinrich Schütze, bin ich dankbar für sein Vertrauen und seine brillianten Ideen (und für den Hinweis, dass es völlig normal ist, wenn Papers abgelehnt werden.) Danke auch an Prof. Dr. Benjamin Roth, dessen Deep-Learning-Vorlesung der Grund ist, warum ich mich für diese Promotion entschieden habe. Ich vermisse unsere gemeinsamen Lehrveranstaltungen!

An meine Mit-Doktorand*innen: Danke für die Unterstützung und für die vielen Diskussionen (akademischer und sonstiger Natur) in Kaffee- und Mittagspausen. Jeder braucht Kolleg*innen wie euch.

An das Team bei Siemens Machine Intelligence, insbesondere Ulli Waltinger und Bernt Andrassy: Danke, dass ihr dieses Projekt finanziert und meiner Dissertation eine Richtung gegeben habt.

An meine Mutter, meine Großeltern, Rudi und Regina: Danke für eure tatkräftige Unterstützung in den letzten Jahren. An Julian: Danke, dass du bei mir bist und mich ermutigst. Du bist der Beste.

Publications and declarations of co-authorship

Chapter 2

Corresponds to the following publication:

> Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020b. Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and Covid-19 QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (presented at SustaiNLP: Workshop on Simple and Efficient Natural Language Processing)*, pages 1482–1490, Online

Declaration of co-authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper and did most of the subsequent corrections. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission. Ulli Waltinger contributed by reviewing an earlier draft of the paper.

Chapter 3

Corresponds to the following publication:

> Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020a. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (presented at Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures)*, pages 803–818, Online

Declaration of co-authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper and did most of the subsequent corrections. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission. Ulli Waltinger contributed by reviewing an earlier draft of the paper.

Chapter 4

Corresponds to the following publication:

> Nina Poerner and Hinrich Schütze. 2019. Multi-view domain adapted sentence embeddings for low-resource unsupervised duplicate question detection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1630–1641, Hong Kong, China

Declaration of co-authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission.

Chapter 5

Corresponds to the following publication:

> Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020c. Sentence meta-embeddings for unsupervised semantic textual similarity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7027–7034, Online

Declaration of co-authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper and did most of the subsequent corrections. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission. Ulli Waltinger contributed by reviewing an earlier draft of the paper.

München, den 10. Dezember 2020

Nina Mareike Pörner

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 About this thesis

1.1.1 Motivation

For much of the last decade, natural language processing (NLP) has been dominated by deep learning: machine learning with complex, parametrized models, which are often referred to as neural networks (see Section 1.2). Some of the most recent advances of the field have been spearheaded by the transfer learning paradigm, where models are pretrained on source tasks, in order to improve performance on target tasks (see Section 1.3).

In the context of NLP, pretrained models can be divided into those that produce **contextualized embeddings**, and those that produce **non-contextualized embeddings**.

Contextualized embeddings (Peters et al., 2017, 2018; Conneau et al., 2017; Howard and Ruder, 2018; Cer et al., 2018; Devlin et al., 2019, inter alia) are vector representations of words and/or sentences that depend on intra-sentential context. For example, the contextualized embedding of the word *bank* is different in the sentence *the bank was robbed* and in the sentence *it is a bank holiday*. Contextualized embeddings are typically produced by deep neural networks like LSTMs (Section 1.4.3.1) or Transformers (Section 1.4.3.2). Since these models are expensive in terms of hardware and training time (Strubell et al., 2019, 2020), pretraining is typically done by big research labs and companies, who then share their checkpoints with the wider NLP community (Wolf et al., 2020).

Non-contextualized embeddings, which are also called static embeddings or simply word vectors, are an older embedding technique (Church and Hanks, 1990; Schütze, 1992; Mikolov et al., 2013c; Pennington et al., 2014; Bojanowski et al., 2017, inter alia). As the name suggests, non-contextualized embeddings do not depend on intra-sentential context, but instead assign one vector per word type. They are usually based on shallow vector operations, and as a result, they are cheaper to pretrain than their contextualized counterparts. But when used for transfer learning purposes, the performance of non-contextualized embeddings may not be comparable to contextualized embeddings.

Imagine an NLP practitioner who wants to tackle some target task via transfer learning. She does not have the resources to pretrain her own contextualized embedding model, so

she decides to use an existing checkpoint. This "off-the-shelf" approach is not optimal in scenarios where there is a mismatch between the source data of the checkpoint and the intended target data (e.g., a domain mismatch, see Section 1.3.2). And while the practitioner has sufficient resources to pretrain non-contextualized embeddings on more suitable source data, the resulting model is unlikely to be comparable to a contextualized one.

1.1.2 Contributions

In this thesis, we explore ways of getting the best of both worlds, by combining contextualized and non-contextualized embeddings. Our contributions can be divided into two avenues:

1.1.2.1 Injecting non-contextualized embeddings into BERT

In Chapters 2 and 3, we propose a method for injecting non-contextualized embeddings into the (contextualized) pretrained BERT model. More specifically, we fit a linear transformation to align non-contextualized embeddings with the vectors of BERT's input layer, based on a small dictionary of shared vocabulary items. This method is inspired by supervised cross-lingual word embedding alignment (Mikolov et al., 2013b; Faruqui and Dyer, 2014; Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017, inter alia). Conceptually, the goal is to "trick" BERT into accepting the transformed non-contextualized embeddings as if they were from its own input layer, without any pretraining of BERT itself.

In Chapter 2, we use this method to inject Word2Vec word embeddings (Mikolov et al., 2013c,a) that were trained on biomedical texts into the general-domain BERT model (Devlin et al., 2019). BERT thus gains knowledge about domain-specific words, with empirical improvements on biomedical named entity recognition and question answering. In Chapter 3, we use the method to inject Wikipedia2Vec entity embeddings (Yamada et al., 2016, 2020) into BERT. The result is a model called E-BERT, which understands a mixture of text and entity inputs. We show empirically that E-BERT is less likely than BERT to over-rely on the surface form of entity names, and that it performs better on unsupervised question answering, relation classification and entity linking tasks.

1.1.2.2 Sentence meta-embeddings

In Chapters 4 and 5, we address unsupervised text similarity tasks with sentence meta-embeddings (called multi-view sentence embeddings in Chapter 4). Sentence meta-embeddings are an application of word meta-embedding techniques (Rastogi et al., 2015; Yin and Schütze, 2016, inter alia) at the level of the sentence. More specifically, we combine ensembles of sentence embeddings from different pretrained models via trainable functions. Empirically, our most successful meta-embedding method is a linear transformation learned with generalized canonical correlation analysis (Kettenring, 1971; Bach and Jordan, 2002).

In Chapter 4, we use this technique to combine contextualized and non-contextualized, domain-specific and general-domain sentence embeddings, in order to perform unsupervised duplicate question detection in highly domain-specific community question answering forums (Hoogeveen et al., 2015; Nakov et al., 2017). In Chapter 5, we use sentence meta-embeddings to set a new state of the art on the unsupervised semantic textual similarity benchmark (Cer et al., 2017).

1.1.3 Outline

Chapters 2 through 5 correspond to the publications that were described in Section 1.1.2. The rest of this introductory chapter provides relevant background information: In Section 1.1.4, we define conventions for mathematical notation. In Sections 1.2 and 1.3, we introduce some core concepts of deep learning and transfer learning. Section 1.4 describes some commonly used neural network architectures for NLP, while Section 1.5 gives an overview of the pretrained models that are used and evaluated in this thesis.

1.1.4 Mathematical notation

The chapters of this thesis were written at different points in time. As a result, they may differ somewhat in terms of mathematical notation. Here, we define some general conventions.

1.1.4.1 Scalars, vectors and matrices

We use lowercase italics for scalars $x \in \mathbb{R}$, lowercase boldface for vectors $\mathbf{x} \in \mathbb{R}^d$, and uppercase boldface for matrices $\mathbf{X} \in \mathbb{R}^{d \times d'}$. The i'th vector of $\mathbf{X}$ is denoted $\mathbf{x}_i$, and the j'th entry of $\mathbf{x}$ is denoted x_j. The inner vector product is denoted $\mathbf{x}^T\mathbf{y}$ or $\mathbf{x} \cdot \mathbf{y}$, while matrix multiplication is denoted $\mathbf{X}\mathbf{y}$. The euclidian norm of $\mathbf{x}$ is $||\mathbf{x}||_2$. Vector concatenation is denoted:

$$[\mathbf{x}; \mathbf{y}] \qquad \text{or} \qquad \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{1.1}$$

1.1.4.2 Functions

Functions are denoted as stylized uppercase $\mathcal{F}$ or lowercase f. We use the notation $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y}$ are the feature space (domain) and output space (codomain) of the function. A composition of functions, where $\mathcal{F}^{(2)}$ is applied after $\mathcal{F}^{(1)}$, is denoted $\mathcal{F}^{(2)} \circ \mathcal{F}^{(1)}$.

1.1.4.3 Texts

Texts are usually written as a sequence of tokens $X = x_1 \dots x_T$, where every token x_t is from a finite vocabulary $\mathbb{L}$ of words, characters, etc. Subsequences of X are denoted $X_{t:t+n}$. The length of X (in tokens) is $|X|$. The size of the vocabulary (in tokens) is $|\mathbb{L}|$.

1.2 Deep learning

Deep learning is machine learning via complex, parametrized models, which are often called neural networks (LeCun et al., 2015; Schmidhuber, 2015, inter alia). Here, we give an overview over the core concepts involved in defining and training a neural network. Throughout this section and the next, we use the task of biomedical named entity recognition as a running example.

1.2.1 Tasks

Neural networks are trained to perform tasks. Following Ruder (2019, p.44), a task is defined by its feature space $\mathcal{X}$ with probability distribution $P(X)$, and its output space $\mathcal{Y}$ with prior distribution $P(Y)$ and conditional probability distribution $P(Y|X)$. In NLP, $\mathcal{X}$ is often a set of texts, i.e., a set of possible sequences of tokens from a vocabulary: $\mathcal{X} = \mathbb{L}^+$, where $+$ is the Kleene plus.

1.2.1.1 Datasets

A task is usually represented by a training set $\mathcal{D}^{(\text{train})}$ and a separate test set $\mathcal{D}^{(\text{test})}$. The former is used to optimize the parameters of the model, while the latter is used to estimate its performance on unseen datapoints.

1.2.1.2 Supervised tasks

Supervised tasks have labeled training data:

$$\begin{aligned} &\mathcal{D}^{(\text{train})} = \{(X^{(1)}, Y^{(1)}), \ldots, (X^{(N)}, Y^{(N)})\} \\ &X^{(n)} \in \mathcal{X}; Y^{(n)} \in \mathcal{Y} \end{aligned} \tag{1.2}$$

Usually, the labels $Y^{(n)}$ stem from annotators or other trusted sources. Tasks with heuristic, less trustworthy labels are often referred to as weakly supervised (Ratner et al., 2020).

Running example: Throughout this section and the next, we use the task of supervised biomedical named entity recognition (NER) as an example.

Here, $\mathcal{X}$ is the space of all sentences, and $P(X)$ is skewed towards sentences from biomedical documents. The task is to predict which subsequences of some sentence $X \in \mathcal{X}$ refer to a certain class of entity, such as diseases. NER is often formalized as a tagging problem, where tokens are classified as *B(egin)*, *(I)nside* or *(O)utside*. Thus, the output space $\mathcal{Y}$ is the space of possible tag sequences: $\mathcal{Y} = \{B, I, O\}^+$. $\mathcal{D}^{(\text{train})}$ and $\mathcal{D}^{(\text{test})}$ contain pairs of sentences and their true tag sequences.

1.2.1.3 Unsupervised tasks

Unsupervised tasks are tasks that use unlabeled data only, i.e., $\mathcal{D}^{(\text{train})} = \{X^{(1)} \dots X^{(N)}\}$. One example of an unsupervised NLP task is autoregressive language modeling, where we predict a given word x_t from its left context $X_{1:t-1}$ or right context $X_{t+1:T}$ (see Section 1.5.2.1). Other unsupervised tasks include problems related to word co-occurrence (see Sections 1.5.1.1 through 1.5.1.3), masked language modeling or next sentence prediction (see Section 1.5.2.2).

1.2.2 Neural networks

1.2.2.1 Layers

A neural network is a function $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ that is parametrized by a set of trainable parameters Θ (Goodfellow et al., 2016, p.164). Section 1.4 describes some commonly used neural network architectures. For now, and regardless of the specific architecture, $\mathcal{F}$ can be conceptualized as a composition of **layers**:

$$\mathcal{F} = \mathcal{F}^{(L)} \circ \dots \circ \mathcal{F}^{(1)} \tag{1.3}$$

Every $\mathcal{F}^{(l)}$ is itself a function with parameters $\Theta^{(l)} \subseteq \Theta$.

1.2.2.2 Embeddings

The outputs of the intermediate layers of a neural network are called hidden vectors. In NLP, hidden vectors are often associated with specific input units, such as words or sentences. These specific vectors are also called **representations** or **embeddings**.

For instance, let $X = x_1 \dots x_T$ be a sentence, and let $\mathbf{H} \in \mathbb{R}^{|X| \times d}$ be the output of an intermediate layer, given X, where the t'th vector $\mathbf{h}_t$ corresponds to the t'th token x_t. Then we would call $\mathbf{h}_t$ an embedding of x_t. As mentioned in Section 1.1.1, we will differentiate between non-contextualized embeddings, where $\mathbf{h}_t$ depends only on x_t, and contextualized embeddings, where $\mathbf{h}_t$ depends on other $x_{t'}$ with $t' \neq t$.

Running example: A neural network that learns our biomedical NER task might look as follows:

Its lowest layer $\mathcal{F}^{(1)}$ is an embedding lookup layer, which transforms the tokens of $X = x_1 \dots x_T$ into non-contextualized embeddings (see Section 1.4.2.1):

$$\mathbf{E} \in \mathbb{R}^{|X| \times d}; \quad \mathbf{E} = \begin{bmatrix} — & \mathbf{e}_1 & — \\ & \vdots & \\ — & \mathbf{e}_T & — \end{bmatrix} = \mathcal{F}^{(1)}(X; \Theta^{(1)}) = \begin{bmatrix} — & \mathbf{w}^{(1)}_{\mathcal{I}_\mathbb{L}(x_1)} & — \\ & \vdots & \\ — & \mathbf{w}^{(1)}_{\mathcal{I}_\mathbb{L}(x_T)} & — \end{bmatrix} \tag{1.4}$$

where $\Theta^{(1)} = \{\mathbf{W}^{(1)} \in \mathbb{R}^{|\mathbb{L}| \times d}\}$, and $\mathcal{I}_\mathbb{L} : \mathbb{L} \to \{1 \dots |\mathbb{L}|\}$ is a bijective indexing function over the vocabulary.

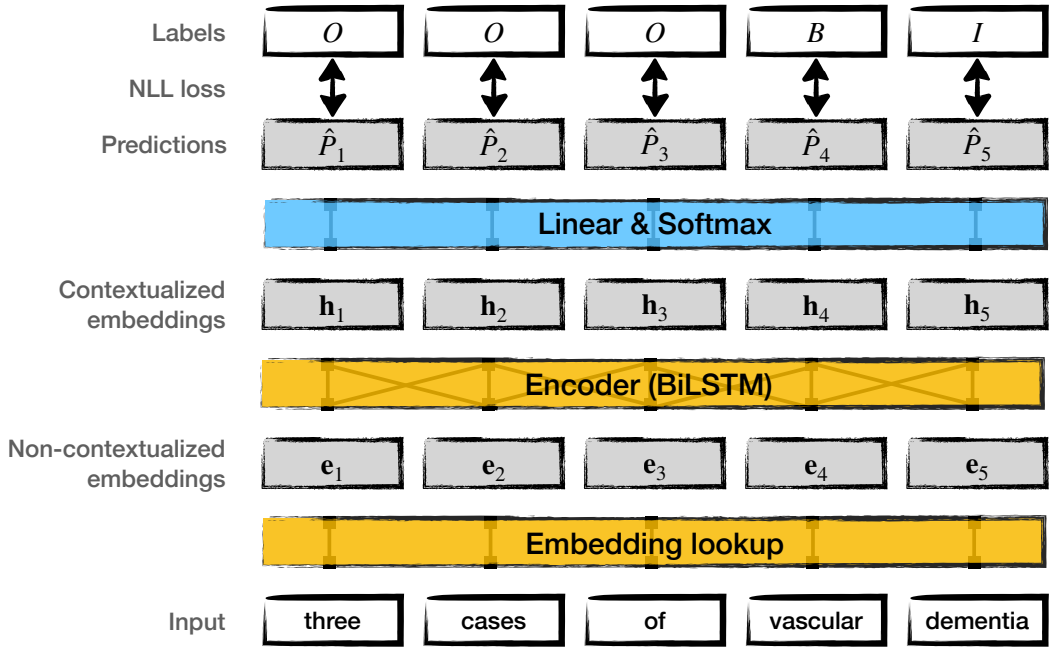

Figure 1.1: Running example: Biomedical NER tagger for diseases.

The second layer $\mathcal{F}^{(2)}$ is a bidirectional LSTM with parameters $\Theta^{(2)}$ (see Section 1.4.3.1), which translates the non-contextualized embeddings into contextualized embeddings. This part of the model is often referred to as the **encoder**:

$$\mathbf{H} \in \mathbb{R}^{|X| \times d'}; \quad \mathbf{H} = \begin{bmatrix} — & \mathbf{h}_1 & — \\ & \vdots & \\ — & \mathbf{h}_T & — \end{bmatrix} = \mathcal{F}^{(2)}(\mathbf{E}; \Theta^{(2)}) \tag{1.5}$$

The third layer $\mathcal{F}^{(3)}$ is a position-wise linear layer with parameters $\Theta^{(3)} = \{\mathbf{W}^{(3)} \in \mathbb{R}^{|\mathcal{Y}| \times d'}, \mathbf{b}^{(3)} \in \mathbb{R}^{|\mathcal{Y}|}\}$. It transforms $\mathbf{H}$ into a matrix of logits (one logit per token and possible NER tag):

$$\mathbf{O} \in \mathbb{R}^{|X| \times |\mathcal{Y}|}; \quad \mathbf{O} = \begin{bmatrix} — & \mathbf{o}_1 & — \\ & \vdots & \\ — & \mathbf{o}_T & — \end{bmatrix} = \mathcal{F}^{(3)}(\mathbf{H}; \Theta^{(3)}); \quad \mathbf{o}_t = \mathbf{W}^{(3)}\mathbf{h}_t + \mathbf{b}^{(3)} \tag{1.6}$$

We apply the softmax function to each $\mathbf{o}_t$, to derive a probability distribution. Let $\mathcal{I}_\mathcal{Y} : \mathcal{Y} \to \{1 \dots |\mathcal{Y}|\}$ be a bijective indexing function over the tags. Then:

$$\hat{P}_t(y|X;\Theta) = \frac{\exp(o_{t,\mathcal{I}_\mathcal{Y}(y)})}{\sum_{y' \in \mathcal{Y}} \exp(o_{t,\mathcal{I}_\mathcal{Y}(y')})} \implies \hat{P}_t(y|X;\Theta) \propto \exp(o_{t,\mathcal{I}_\mathcal{Y}(y)}) \tag{1.7}$$

where $\Theta = \Theta^{(1)} \cup \Theta^{(2)} \cup \Theta^{(3)}$. At test time, we would choose the tag that maximizes $\hat{P}_t$:

$$\hat{y}_t = \operatorname*{argmax}_{y \in \mathcal{Y}} \hat{P}_t(y|X;\Theta) \tag{1.8}$$

1.2.2.3 Loss functions

During training, the goal is to find the parameters Θ that minimize a loss $\mathcal{L}(\Theta)$. Usually, $\mathcal{L}$ is the sum or average of a datapoint-level loss function l over the training set:

$$\mathcal{L}(\Theta) = \frac{1}{|\mathcal{D}^{(\text{train})}|} \sum_{(X,Y)\in\mathcal{D}^{(\text{train})}} l(\Theta; X, Y) \tag{1.9}$$

The exact formulation of l depends on the task at hand. For a regression task, where $\mathcal{Y} = \mathbb{R}$, it might be the squared error of the predicted and true scalars:

$$l(\Theta; X, Y) = (\mathcal{F}(X; \Theta) - Y)^2 \tag{1.10}$$

For a classification task, where $\mathcal{Y}$ is a set of discrete labels (categories), the loss might be the negative log likelihood (NLL) of the true label. In our NER example, a typical loss function would be the average negative log likelihood of the true tags:

$$l(\Theta; X, Y) = -\frac{1}{|X|} \sum_{t=1}^{|X|} \ln \hat{P}_t(y_t|X; \Theta) \tag{1.11}$$

1.2.2.4 Gradient descent

The loss is usually minimized via gradient descent. Let $\nabla_\theta \mathcal{L}(\Theta)$ denote the gradient of the loss with respect to some parameter $\theta \in \Theta$, and let $\eta > 0$ be our learning rate. Then the gradient update would be:

$$\begin{aligned} \theta^{(\text{new})} &= \theta^{(\text{old})} - \eta \nabla_\theta \mathcal{L}(\Theta) \\ &= \theta^{(\text{old})} - \frac{\eta}{|\mathcal{D}^{(\text{train})}|} \sum_{(X,Y)\in\mathcal{D}^{(\text{train})}} \nabla_\theta l(\Theta; X, Y) \end{aligned} \tag{1.12}$$

In practice, updates are not normally performed on the entire training set. Instead, *mini-batch gradient descent* calculates each update on a batch $\mathcal{B}^{(\text{train})}$, which contains a few dozen or hundred datapoints from $\mathcal{D}^{(\text{train})}$:

$$\theta^{(\text{new})} = \theta^{(\text{old})} - \frac{\eta}{|\mathcal{B}^{(\text{train})}|} \sum_{(X,Y)\in\mathcal{B}^{(\text{train})}} \nabla_\theta l(\Theta; X, Y) \tag{1.13}$$

The special case where every batch contains a single datapoint (i.e., $\mathcal{B}^{(\text{train})} = \{(X, Y)\}$) is called *stochastic gradient descent* (Bottou, 2010):

$$\theta^{(\text{new})} = \theta^{(\text{old})} - \eta \nabla_\theta l(\Theta; X, Y) \tag{1.14}$$

1.2.3 The problem of data sparsity

In a single-task setup, the parameters Θ are trained exclusively on $\mathcal{D}^{(\text{train})}$. This means that any information that is not present in $\mathcal{D}^{(\text{train})}$ cannot be learned. This may lead to poor performance on the test data, especially if $\mathcal{D}^{(\text{train})}$ is small.

Running example: Assume that the test set of our biomedical NER task contains a sentence with the word *VaD*, which is an abbreviation for *vascular dementia*. Assume further that while *vascular dementia* has occurred in $\mathcal{D}^{(\text{train})}$, *VaD* has not. A single-task model may therefore be unable to tag *VaD* as a disease entity.

1.3 Transfer learning

In this section, we introduce the transfer learning paradigm. Transfer learning addresses data sparsity by transferring knowledge between tasks (Pan and Yang, 2009; Ruder, 2019, inter alia). A transfer learning scenario has at least two tasks: a target task $\mathcal{T}_T$, which is the task that we are actually interested in, and a source task $\mathcal{T}_S$, which is used to improve performance on $\mathcal{T}_T$.

Transfer learning scenarios can be characterized by the ways in which the source and target tasks differ. Following Ruder (2019, p.44), we distinguish between scenarios where the mismatch is between the tasks themselves (i.e., their output spaces) and scenarios where the mismatch is between their domains (i.e., their feature spaces or feature probability distributions).

1.3.1 Task mismatches

Task mismatches occur when $\mathcal{Y}_S \neq \mathcal{Y}_T$. For instance, $\mathcal{T}_S$ may be a part-of-speech tagging task with $\mathcal{Y}_S = \{NOUN, VERB, \ldots\}^+$, while $\mathcal{T}_T$ is an NER tagging task with $\mathcal{Y}_T = \{B, I, O\}^+$. Ruder (2019) further differentiates between multi-task transfer learning, where $\mathcal{T}_S$ and $\mathcal{T}_T$ are learned simultaneously, and sequential transfer learning, where $\mathcal{T}_S$ is used to pretrain some (or all) layers the model used for $\mathcal{T}_T$. Since this thesis does not cover multi-task scenarios, we only describe sequential transfer learning.

1.3.1.1 Sequential transfer learning

With sequential transfer learning, some (or all) layers of the target model $\mathcal{F}_T$ are transferred from a source model $\mathcal{F}_S$. Let $\mathcal{F}_S = \mathcal{F}_S^{(L)} \circ \ldots \circ \mathcal{F}_S^{(1)}$ with parameters Θ_S be a model that was pretrained on the training set $\mathcal{D}_S^{(\text{train})}$ of a source task $\mathcal{T}_S$ (or several source tasks).

Let $(\mathcal{F}_S^{(M)} \ldots \mathcal{F}_S^{(N)})$, with parameters $\bar{\Theta}_S \subseteq \Theta_S$, be a list of layers designated for transfer. Usually, these layers are contiguous, i.e., they can be written as:

$$\bar{\mathcal{F}}_S = \mathcal{F}_S^{(N)} \circ \ldots \circ \mathcal{F}_S^{(M)} \tag{1.15}$$

After pretraining, $\bar{\mathcal{F}}_S$ becomes a building block of $\mathcal{F}_T$. The second building block are normally some task-specific layers, denoted $\bar{\mathcal{F}}_T$, with randomly initialized parameters $\bar{\Theta}_T$. Usually, $\bar{\mathcal{F}}_S$ forms the lower layer(s) of $\mathcal{F}_T$, while $\bar{\mathcal{F}}_T$ forms the upper layer(s):

$$\mathcal{F}_T = \bar{\mathcal{F}}_T \circ \bar{\mathcal{F}}_S \tag{1.16}$$

One way to conceptualize this model is to say that $\bar{\mathcal{F}}_S$ produces pretrained **embeddings** (contextualized or not) for $\bar{\mathcal{F}}_T$. Depending on the architecture, $\bar{\mathcal{F}}_S$ may be as simple as an embedding lookup layer (see Figure 1.2a), or as complex as a deep stack of Transformer blocks (see Figure 1.2b). The depth of $\bar{\mathcal{F}}_T$ will usually be chosen based on whether $\bar{\mathcal{F}}_S$ is complex enough to do the heavy lifting during target task training.

Zero-shot transfer: A special case of sequential transfer learning arises when $\mathcal{T}_T$ is a zero-shot task without a training set $\mathcal{D}_T^{(\text{train})}$. This includes target tasks that are reformulated as a version of the source task (Radford et al., 2019; Petroni et al., 2019; Brown et al., 2020, inter alia), or unsupervised text similarity tasks that are formulated as an embedding similarity problem (see Chapters 4, 5). In these cases, there are usually no task-specific layers, i.e., $\mathcal{F}_T = \bar{\mathcal{F}}_S$ (see Figure 1.2c).

Freezing versus finetuning: If a training set $\mathcal{D}_T^{(\text{train})}$ exists, the task-specific parameters $\bar{\Theta}_T$ are usually trained on it after transfer. An important design choice is whether to "freeze" the pretrained parameters $\bar{\Theta}_S$, or to train ("finetune") them along with $\bar{\Theta}_T$. With contextualized embeddings, finetuning is usually reported to be more successful (Devlin et al., 2019); with non-contextualized embeddings, both options are popular.

Representation learning: The special case where $\mathcal{T}_S$ is an unsupervised task is often referred to as representation learning (Bengio et al., 2013). Recently, the distinction between representation learning and supervised sequential transfer learning has been blurred by models that are pretrained on a mix of supervised and unsupervised data (Conneau et al., 2017; Cer et al., 2018; Liu et al., 2019a, inter alia). Furthermore, some training or retrofitting methods for non-contextualized embeddings are supervised as well (Faruqui et al., 2015; Rothe et al., 2016; Glavaš and Vulić, 2018; Dufter and Schütze, 2019, inter alia). In this thesis, we will not usually make a distinction between representation learning and supervised transfer learning. Instead, we will use the terms **representations** or **embeddings** to refer to the outputs of pretrained models, regardless of whether they were pretrained on supervised or unsupervised source tasks (or both).

Running example: To illustrate the usefulness of sequential transfer learning, we return to our running example. This time, assume that we have access to a language model $\mathcal{F}_S$, as described in Section 1.2.1.3, which was pretrained on a large corpus of unlabeled text data. Let $\bar{\mathcal{F}}_S$ be the embedding lookup and encoder layers of $\mathcal{F}_S$, which produce a matrix of contextualized word embeddings $\mathbf{H} \in \mathbb{R}^{|X| \times d'}$. We plug $\mathbf{H}$ into Equation 1.6, effectively replacing the two lower layers of $\mathcal{F}_T$ (yellow in Figure 1.1).

Now, assume that when $\mathcal{F}_S$ was pretrained, it learned that the phrase *vascular dementia* and its abbreviation *VaD* occur in similar contexts. Thus, the lower layers $\bar{\mathcal{F}}_S$ learned to assign similar contextualized embeddings to occurrences of *vascular dementia* and *VaD*. So when the task-specific final linear layer $\bar{\mathcal{F}}_T$ (blue in Figure 1.1) learns to tag *vascular dementia* as a disease, it has a better chance of correctly tagging *VaD* as well.

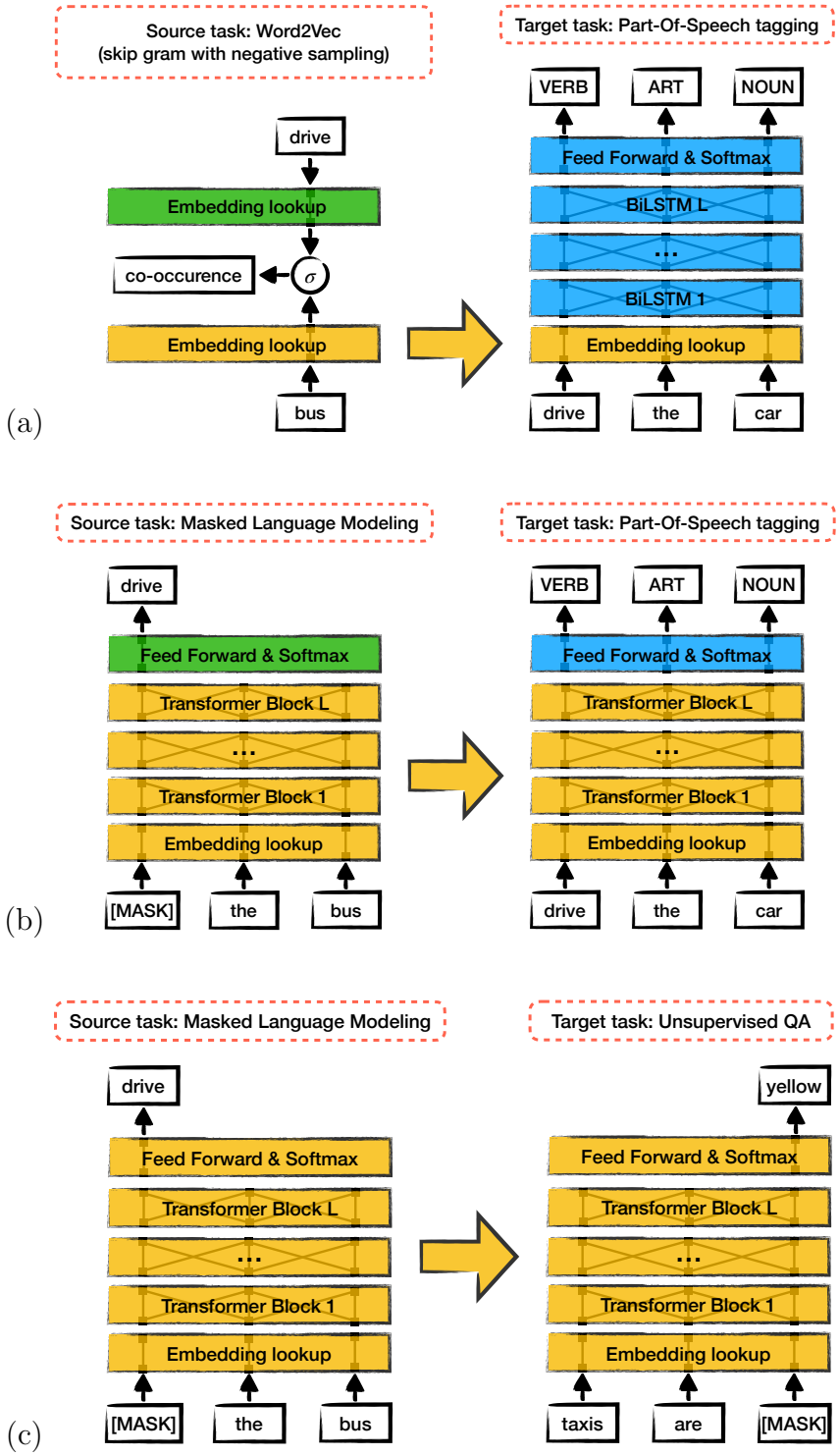

Figure 1.2: Examples of sequential transfer learning (schematic). Yellow: $\bar{\mathcal{F}}_S$. Blue: $\bar{\mathcal{F}}_T$. Green: Other layers in $\mathcal{F}_S$. (a) Deep task-specific BiLSTM on top of pretrained non-contextualized Word2Vec embeddings. (b) Shallow task-specific classifier on top of pretrained contextualized BERT layers. (c) Pretrained BERT model used for a zero-shot target task, without any task-specific layers.

1.3.2 Domain mismatches

Domain mismatches are situations where the source and target tasks differ in terms of their input data. Following Ruder (2019), we further differentiate between situations where $\mathcal{X}_S \neq \mathcal{X}_T$ (called feature space mismatches below) and situations where $P_S(X_S) \neq P_T(X_T)$ (domain mismatches in the narrow sense).

1.3.2.1 Feature space mismatches

In NLP, feature space mismatches typically occur when $\mathcal{X}_S$ and $\mathcal{X}_T$ are defined over different vocabularies $\mathbb{L}_S \neq \mathbb{L}_T$. According to Ruder (2019)'s taxonomy, this issue mainly arises in cross-lingual transfer. In Chapter 3, we address a monolingual scenario where $\mathbb{L}_S$ contains wordpiece tokens (i.e., words and subwords), while $\mathbb{L}_T \supset \mathbb{L}_S$ contains wordpiece and entity tokens.

1.3.2.2 Domain mismatches (narrow sense)

Domain mismatches in the narrow sense are situations where $P_S(X_S)$ and $P_T(X_T)$ represent different types of text. For instance, the source data might contain text from Wikipedia, while the target data contains text from biomedical publications. Methods that deal with domain mismatch are usually called **domain adaptation**. We address domain mismatch problems in Chapters 2 and 4.

1.3.3 Task and domain mismatches combined

Of course, task mismatches and domain mismatches are not mutually exclusive. For instance, a neural network that was pretrained as a language model on Wikipedia is doubly-mismatched for biomedical NER: It does not know the target domain (biomedical publications), and it does not know the target task (NER tagging).

Often, these double-mismatches are addressed via a two-step procedure, where a model first undergoes additional pretraining on more suitable data, followed by finetuning on the target task (Howard and Ruder, 2018; Beltagy et al., 2019; Alsentzer et al., 2019; Peng et al., 2019; Lee et al., 2020, inter alia). But since contextualized embedding models are expensive to pretrain, this approach may not always be practical.

1.4 Neural network architectures for NLP

In this section, we describe some commonly used neural network architectures for NLP. First, we briefly discuss tokenization methods.

characters: *c, h, e, s, t, _, p, a, i, n, _, a, n, d, _, t, a, c, h, y, c, a, r, d, i, a*

words: *chest, pain, and, [UNK]* (assume that *tachycardia* $\notin \mathbb{L}$)

wordpieces (BERT): *chest, pain, and, ta, ##chy, ##card, ##ia*

sentencepieces (T5): *_chest, _pain, _and, _, t, a, chy, cardi, a*

Figure 1.3: Four ways of tokenizing the string *chest pain and tachycardia*. _ stands for the white-space character, ## signals a non-initial wordpiece.

1.4.1 Tokenization

Let $\mathbb{C}$ be a vocabulary of characters (or bytes), and let $\mathbb{C}^+$ be the set of all possible strings. Tokenization can be formalized as a function $\mathcal{Z} : \mathbb{C}^+ \to \mathbb{L}^+$, where $\mathbb{L}$ is a finite set of so-called "tokens". These tokens are the atomic units of input to the lowest layer of the neural network.

The simplest way of achieving tokenization is to perform character (or byte) tokenization, by defining $\mathcal{Z}(X) = X$. Character tokenization breaks up semantically meaningful units (such as words), and places the burden of reconstructing their meaning onto the downstream model. Alternatively, one can use a rule-based word tokenizer (e.g., a white-space and punctuation tokenizer) and define $\mathbb{L}$ to be the top-k most frequent words in the training set. Problems with word tokenization include the inability to represent unknown words $x \notin \mathbb{L}$, and the need for language-specific rule engineering (e.g., deciding what counts as a punctuation mark, how to tokenize abbreviations, etc.).

Wordpiece (Wu et al., 2016) and sentencepiece (Kudo and Richardson, 2018) tokenization provide a middle ground between character tokenization and word tokenization. Both are data-driven algorithms that build a vocabulary of subwords ("pieces") from unlabeled text. Frequent words are represented as a single piece, while less frequent words are tokenized (see Figure 1.3).

1.4.2 Architectures for non-contextualized embeddings

1.4.2.1 Word vectors

The simplest form of non-contextualized embeddings are word vectors. Let $\mathcal{I}_{\mathbb{L}} : \mathbb{L} \to \{1 \dots |\mathbb{L}|\}$ be a bijective indexing function, and let $\mathbf{W} \in \mathbb{R}^{|\mathbb{L}| \times d}$ be a parameter matrix. The word vector of $x \in \mathbb{L}$ is defined as $\mathbf{e} = \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x)}$. To embed a tokenized text $X = x_1 \dots x_T$, we simply stack the vectors of the tokens:

$$\mathbf{E} = \begin{bmatrix} — & \mathbf{e}_1 & — \\ & \vdots & \\ — & \mathbf{e}_T & — \end{bmatrix} = \begin{bmatrix} — & \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x_1)} & — \\ & \vdots & \\ — & \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x_T)} & — \end{bmatrix}$$

Note that non-contextualized embeddings can be more complex than atomic word vectors. For instance, FastText (Bojanowski et al., 2017, see Section 1.5.1.2) calculates word embeddings as an average of word and subword vectors. The resulting embeddings are still non-contextualized (at the sentence level), since the subwords of some word x_t in sentence X do not depend on any other words $x_{t'}$, where $t' \neq t$.

1.4.2.2 Sentence embeddings from non-contextualized embeddings

Assume that we want to represent a sentence $X = x_1 \dots x_T$ as a single sentence embedding $\mathbf{s} \in \mathbb{R}^d$. A simple way of achieving this is to take a weighted average over the non-contextualized embeddings of its tokens:

$$\mathbf{s} = \frac{1}{|X|} \sum_{t=1}^{|X|} \mathcal{A}(x_t)\mathbf{e}_t = \frac{1}{|X|} \sum_{t=1}^{|X|} \mathcal{A}(x_t)\mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x_t)}$$

$\mathcal{A} : \mathbb{L} \rightarrow \mathbb{R}$ is an optional weighting function, which is usually chosen to down-weight frequent word types (Arora et al., 2017; Ethayarajh, 2018). Despite their simplicity and their inability to encode word order, averaged non-contextualized word embeddings are surprisingly effective sentence embeddings for some tasks (Conneau and Kiela, 2018). We use them as part of our sentence meta-embeddings in Chapters 4 and 5.

1.4.3 Architectures for contextualized embeddings

1.4.3.1 LSTM

Long Short-Term Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are recurrent neural networks. Given some sentence $X = x_1 \dots x_T$, they usually produce one contextualized embedding $\mathbf{h}_t$ per token x_t.

Let $\mathbf{E} \in \mathbb{R}^{|X| \times d}$ be a matrix of embeddings associated with the tokens of X (e.g., a matrix of non-contextualized embeddings as described in Section 1.4.2.1). Let $\Theta = \{\mathbf{W}^{(i)}, \mathbf{W}^{(f)}, \mathbf{W}^{(o)}, \mathbf{W}^{(g)} \in \mathbb{R}^{d' \times d}, \mathbf{U}^{(i)}, \mathbf{U}^{(f)}, \mathbf{U}^{(o)}, \mathbf{U}^{(g)} \in \mathbb{R}^{d' \times d'}, \mathbf{b}^{(i)}, \mathbf{b}^{(f)}, \mathbf{b}^{(o)}, \mathbf{b}^{(g)} \in \mathbb{R}^{d'}\}$ be parameters. The LSTM architecture is recursively defined as:

$$\begin{aligned}
\mathbf{h}_0 &= \mathbf{0} \\
\mathbf{c}_0 &= \mathbf{0} \\
\mathbf{i}_t &= \sigma(\mathbf{W}^{(i)}\mathbf{e}_t + \mathbf{U}^{(i)}\mathbf{h}_{t-1} + \mathbf{b}^{(i)}) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^{(f)}\mathbf{e}_t + \mathbf{U}^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^{(o)}\mathbf{e}_t + \mathbf{U}^{(o)}\mathbf{h}_{t-1} + \mathbf{b}^{(o)}) \\
\mathbf{g}_t &= \tanh(\mathbf{W}^{(g)}\mathbf{e}_t + \mathbf{U}^{(g)}\mathbf{h}_{t-1} + \mathbf{b}^{(g)}) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1.17}$$

where σ and tanh are the elementwise sigmoid and hyperbolic tangent functions, and $\odot$ is the Hadamard product. Conceptually, $\mathbf{c}_t$ is the model's long term memory. The input gate $\mathbf{i}_t$ controls what information is written into $\mathbf{c}_t$, while the forget gate $\mathbf{f}_t$ controls what information is discarded. The output gate $\mathbf{o}_t$ controls what information from $\mathbf{c}_t$ is exposed to $\mathbf{h}_t$.

Often, LSTMs are defined bidirectionally. A bidirectional LSTM (BiLSTM) consists of two LSTMs with separate sets of parameters $\overrightarrow{\Theta}, \overleftarrow{\Theta}$. The first LSTM reads the input left-to-right, while the other reads it right-to-left. With a BiLSTM, the contextualized embedding of x_t is usually defined as $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$.

1.4.3.2 Transformer

A more recent successful architecture is the Transformer (Vaswani et al., 2017). In contrast to LSTMs, Transformers do not have an inductive bias towards sequential processing. Also, Transformers are not (inherently) autoregressive and therefore easy to parallelize.

Transformers are stacks of "blocks". Let $X = x_1 \dots x_T$ be our input, i.e., a sequence of tokens. Each block $\mathcal{F}^{(l)}$ receives as input a matrix of token embeddings $\mathbf{H}^{(l-1)} \in \mathbb{R}^{|X| \times d}$, and outputs a new matrix $\mathbf{H}^{(l)} \in \mathbb{R}^{|X| \times d}$. Each block consists of a multi-head self-attention layer and a feed-forward layer, which are surrounded by layer normalization and residual connections. Here, we describe each component in detail.

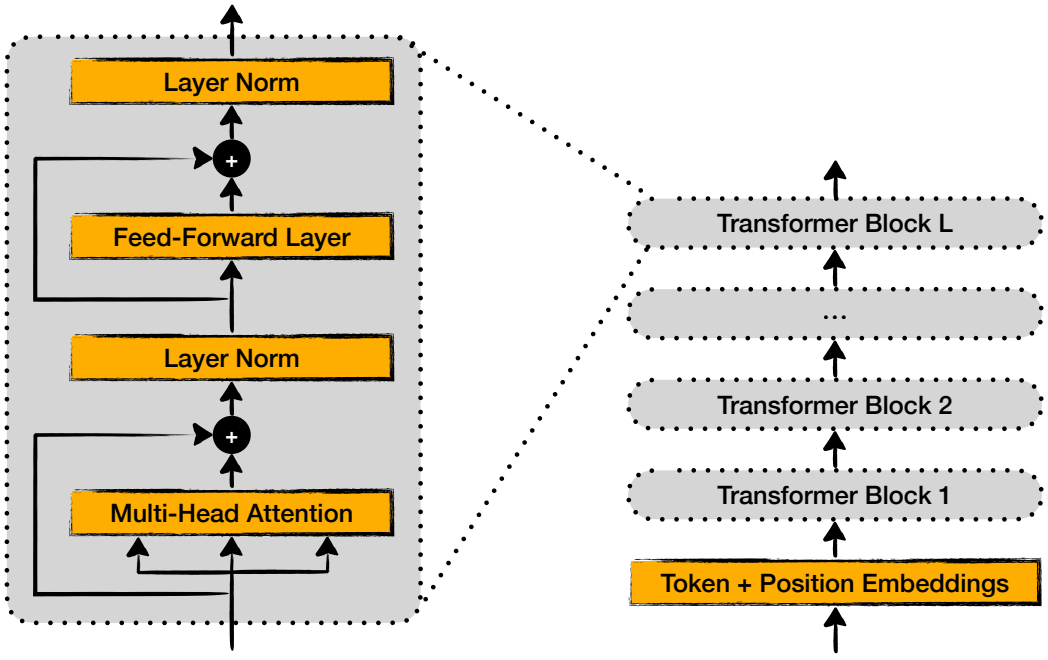

Figure 1.4: Transformer architecture (encoder only), following Figure 1 in Vaswani et al. (2017).

Self-attention: Self-attention layers (Lin et al., 2017) model the interactions of tokens via an attention matrix. Here, we describe scaled dot product self-attention (Vaswani et al., 2017): Let $\Theta^{(\text{attn})} = \{\mathbf{W}^{(q)}, \mathbf{W}^{(k)} \in \mathbb{R}^{d \times d_k}, \mathbf{W}^{(v)} \in \mathbb{R}^{d \times d_v}\}$. The first step is to transform $\mathbf{H}$ into query, key and value matrices $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{|X| \times d_k}, \mathbf{V} \in \mathbb{R}^{|X| \times d_v}$:

$$\begin{aligned} \mathbf{Q} &= \mathbf{H}\mathbf{W}^{(q)} \\ \mathbf{K} &= \mathbf{H}\mathbf{W}^{(k)} \\ \mathbf{V} &= \mathbf{H}\mathbf{W}^{(v)} \end{aligned} \tag{1.18}$$

The attention matrix $\mathbf{A} \in (0,1)^{|X| \times |X|}$ is defined by the softmax function, taken over the scaled dot products of query and key vectors. Often, $a_{t,t'}$ is interpreted as the "attention" that x_t pays to $x_{t'}$.

$$a_{t,t'} = \frac{\exp(\frac{\mathbf{q}_t \cdot \mathbf{k}_{t'}}{\sqrt{d_k}})}{\sum_{t''=1}^{|X|} \exp(\frac{\mathbf{q}_t \cdot \mathbf{k}_{t''}}{\sqrt{d_k}})} \tag{1.19}$$

The final step is a simple attention-weighted sum over the value vectors:

$$\mathcal{F}^{(\text{attn})}(\mathbf{H}; \Theta^{(\text{attn})}) = \mathbf{A}\mathbf{V}$$

Multi-head self-attention: Multi-head self-attention is the application of several self-attention layers ("heads") in parallel. Let $(\mathcal{F}_1^{(\text{attn})} \ldots \mathcal{F}_M^{(\text{attn})})$ be self-attention layers with separate parameters $(\Theta_1^{(\text{attn})} \ldots \Theta_M^{(\text{attn})})$. Let $\Theta^{(\text{mha})} = \{\mathbf{W}^{(o)} \in \mathbb{R}^{Md_v \times d}\} \cup \Theta_1^{\text{attn}} \cup \ldots \cup \Theta_M^{(\text{attn})}$. The multi-head self-attention layer is defined as the concatenation of all heads, down-projected by $\mathbf{W}^{(o)}$:

$$\mathcal{F}^{(\text{mha})}(\mathbf{H}; \Theta^{(\text{mha})}) = [\mathcal{F}_1^{(\text{attn})}(\mathbf{H}; \Theta_1^{(\text{attn})}); \ldots; \mathcal{F}_M^{(\text{attn})}(\mathbf{H}; \Theta_M^{(\text{attn})})]\mathbf{W}^{(o)} \tag{1.20}$$

Feed-forward layer: The feed-forward layer consists of two linear layers, with an elementwise non-linearity $\mathcal{G}$ in between. It is applied position-wise. Let $\Theta^{(\text{ff})} = \{\mathbf{W}^{(1)} \in \mathbb{R}^{d' \times d}, \mathbf{W}^{(2)} \in \mathbb{R}^{d \times d'}, \mathbf{b}^{(1)} \in \mathbb{R}^{d'}, \mathbf{b}^{(2)} \in \mathbb{R}^{d}\}$ be parameters. Then:

$$\mathcal{F}^{(\text{ff})}(\mathbf{H}; \Theta^{(\text{ff})}) = \begin{bmatrix} \text{——} & \mathbf{W}^{(2)}\mathcal{G}(\mathbf{W}^{(1)}\mathbf{h}_1 + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} & \text{——} \\ & \vdots & \\ \text{——} & \mathbf{W}^{(2)}\mathcal{G}(\mathbf{W}^{(1)}\mathbf{h}_T + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} & \text{——} \end{bmatrix} \tag{1.21}$$

Residual connection and layer normalization: Every multi-head self-attention layer and every feed-forward layer is wrapped with a residual connection, followed by position-wise layer normalization (Ba et al., 2016). Let $\Theta^{(\text{norm})} = \{\boldsymbol{\gamma} \in \mathbb{R}^d, \boldsymbol{\beta} \in \mathbb{R}^d\}$. Then layer normalization is defined as:

$$\mathcal{F}^{(\text{norm})}(\mathbf{H}; \Theta^{(\text{norm})}) = \begin{bmatrix} \text{——} & \boldsymbol{\gamma} \odot \frac{\mathbf{h}_1 - \mu_1}{s_1} + \boldsymbol{\beta} & \text{——} \\ & \vdots & \\ \text{——} & \boldsymbol{\gamma} \odot \frac{\mathbf{h}_T - \mu_T}{s_T} + \boldsymbol{\beta} & \text{——} \end{bmatrix} \tag{1.22}$$

where

$$\mu_t = \frac{1}{d}\sum_{i=1}^{d} h_{t,i}; \qquad s_t = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(h_{t,i}-\mu_t)^2} \tag{1.23}$$

Let $\mathcal{F}^{(\text{norm-mha})}, \mathcal{F}^{(\text{norm-ff})}$ be two separate layer normalization layers. Then the l'th Transformer block is defined as:

$$\begin{aligned}\mathbf{H}'^{(l)} &= \mathcal{F}^{(\text{norm-mha})}(\mathcal{F}^{(\text{mha})}(\mathbf{H}^{(l-1)};\Theta^{(\text{mha})}) + \mathbf{H}^{(l-1)};\Theta^{(\text{norm-mha})}) \\ \mathbf{H}^{(l)} &= \mathcal{F}^{(l)}(\mathbf{H}^{(l-1)};\Theta^{(l)}) = \mathcal{F}^{(\text{norm-ff})}(\mathcal{F}^{(\text{ff})}(\mathbf{H}'^{(l)};\Theta^{(\text{ff})}) + \mathbf{H}'^{(l)};\Theta^{(\text{norm-ff})})\end{aligned} \tag{1.24}$$

where $\Theta^{(l)} = \Theta^{(\text{mha})} \cup \Theta^{(\text{ff})} \cup \Theta^{(\text{norm-mha})} \cup \Theta^{(\text{norm-ff})}$.

Input embeddings: The input to the first Transformer block, $\mathbf{H}^{(0)}$, is a matrix of non-contextualized embeddings from an embedding lookup layer with parameter $\mathbf{W} \in \mathbb{R}^{|\mathbb{L}|\times d}$, as described in Section 1.4.2.1, For most pretrained Transformers, $\mathbb{L}$ is a moderately-sized vocabulary of wordpieces or sentencepieces (see Section 1.4.1).

Vanilla self-attention is unable to differentiate between two sentences that contain the same non-contextualized embeddings in a different word order. Hence, information about position-in-sentence is usually injected by position embeddings. Let $\mathbf{P} \in \mathbb{R}^{|X|\times d}$ be a lookup parameter matrix for position embeddings, which may be trainable (Devlin et al., 2019) or deterministic (Vaswani et al., 2017). Then the input to the first block is:

$$\mathbf{H}^{(0)} = \mathcal{F}^{(0)}(X;\Theta^{(0)}) = \begin{bmatrix} \text{——} & \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x_1)} + \mathbf{p}_1 & \text{——} \\ & \vdots & \\ \text{——} & \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x_T)} + \mathbf{p}_T & \text{——} \end{bmatrix} \tag{1.25}$$

Some models add additional embeddings to $\mathbf{H}^{(0)}$. For instance, BERT has an additional parameter matrix $\mathbf{G} \in \mathbb{R}^{2\times d}$, which signals the difference between two texts (see Section 1.5.2.2). The set of trainable parameters used inside $\mathcal{F}^{(0)}$ is denoted $\Theta^{(0)}$.

Putting it all together: Let $\Theta^{(\text{tf})} = \Theta^{(0)} \cup \Theta^{(1)} \cup \ldots \cup \Theta^{(L)}$. The full Transformer architecture is defined as:

$$\mathcal{F}^{(\text{tf})}(X;\Theta^{(\text{tf})}) = (\mathcal{F}^{(L)} \circ \ldots \circ \mathcal{F}^{(1)} \circ \mathcal{F}^{(0)})(X;\Theta^{(L)} \ldots \Theta^{(1)}, \Theta^{(0)}) \tag{1.26}$$

See Figure 1.4 for a depiction of the architecture.

1.5 Pretrained models

In this section, we introduce some popular pretrained models, which we use and evaluate in this thesis. For our purpose, a pretrained model is characterized by its source task (or source tasks) $\mathcal{T}_S$ with dataset $\mathcal{D}_S^{(\text{train})}$, a source architecture $\mathcal{F}_S$ with parameters Θ_S, and a transferred architecture $\bar{\mathcal{F}}_S$ with parameters $\bar{\Theta}_S$. Usually, $\bar{\Theta}_S \subseteq \Theta_S$.

	Architecture	output embeddings	contextualized?
Word2Vec	embedding lookup	word embeddings	no
FastText	embedding lookup & average	word embeddings	no
GloVe	embedding lookup	word embeddings	no
Wikipedia2Vec	embedding lookup	word & entity embeddings	no
ParaNMT	embedding lookup & average	sentence embeddings	no
ELMo	character CNN & BiLSTM	word embeddings	yes
BERT	Transformer	wordpiece & sentence embeddings	yes
InferSent	BiLSTM	sentence embeddings	yes
USE	Transformer	sentence embeddings	yes
SBERT	Transformer	sentence embeddings	yes

	Source task(s)	supervised?
Word2Vec	classification: word co-occurrence	no
FastText	classification: word co-occurrence	no
GloVe	regression: word co-occurrence	no
Wikipedia2Vec	see Word2Vec (above) & classification: Wikipedia links	partially
ParaNMT	classification: synthetic paraphrases	weakly
ELMo	autoregressive language modeling	no
BERT	masked language modeling & next sentence prediction	no
InferSent	see GloVe (above) & natural language inference	partially
USE	skip-thought & conversation & natural language inference	partially
SBERT	see BERT (above) & natural language inference	partially

	citation	described in	used in
Word2Vec	Mikolov et al. (2013c)	Section 1.5.1.1	Chapter 2
FastText	Bojanowski et al. (2017)	Section 1.5.1.2	Chapter 4
GloVe	Pennington et al. (2014)	Section 1.5.1.3	Chapter 4
Wikipedia2Vec	Yamada et al. (2016)	Section 1.5.1.4	Chapter 3
ParaNMT	Wieting and Gimpel (2018)	Section 1.5.1.5	Chapters 4 and 5
ELMo	Peters et al. (2018)	Section 1.5.2.1	Chapter 4
BERT	Devlin et al. (2019)	Section 1.5.2.2	Chapters 4, 2 and 3
InferSent	Conneau et al. (2017)	Section 1.5.2.3	Chapter 4
USE	Cer et al. (2018)	Section 1.5.2.4	Chapters 4 and 5
SBERT	Reimers and Gurevych (2019)	Section 1.5.2.5	Chapter 5

Table 1.1: Summary of pretrained models that are used in this thesis.

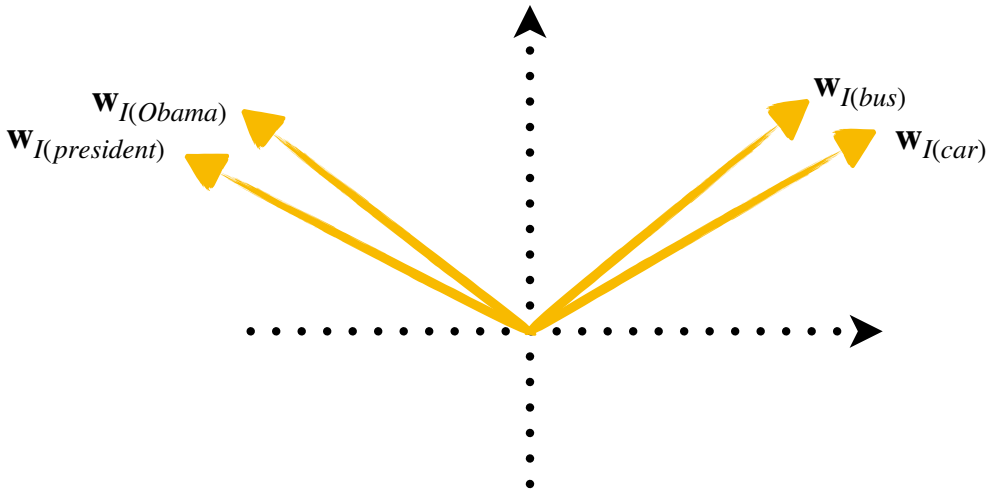

Figure 1.5: Schematic depiction of pretrained Word2Vec vectors: Words that are semantically similar (that appear in similar textual contexts) get similar word vectors.

Depending on the architecture, pretrained models differ with respect to the type of embeddings that they return (e.g., sentence embeddings, word(-piece) embeddings), and whether the embeddings are contextualized. The source task determines whether a given pretrained model is classified as supervised or unsupervised. Table 1.1 summarizes the models described in this section.

1.5.1 Pretrained models for non-contextualized embeddings

1.5.1.1 Word2Vec

Word2Vec (Mikolov et al., 2013a,c) is a model for non-contextualized word embeddings. In Chapter 2, we pretrain Word2Vec on biomedical texts, and inject the resulting embeddings into BERT. Here, we describe the skip-gram negative sampling variant of Word2Vec.

Let $\mathbb{L}$ be a finite vocabulary of word types, and let $\mathcal{I}_{\mathbb{L}}$ be the associated indexing function. Let $\bar{\Theta}_S = \{\mathbf{W} \in \mathbb{R}^{|\mathbb{L}| \times d}\}$. When using Word2Vec for transfer learning, one would typically initialize the rows of the word embedding lookup layer of a target model with:

$$\bar{\mathcal{F}}_S(x; \bar{\Theta}_S) = \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x)} \tag{1.27}$$

During pretraining, Word2Vec predicts whether a given word pair $x, x' \in \mathbb{L}$ is a true co-occurrence in the unsupervised training set $\mathcal{D}_S^{(\text{train})}$, or a random word pair (negative sample). Let $\Theta_S = \bar{\Theta}_S \cup \{\mathbf{V}\}$, where $\mathbf{V} \in \mathbb{R}^{|\mathbb{L}| \times d}$ is another parameter matrix. Then the score of the word pair x, x' is:

$$\mathcal{F}_S(x, x'; \Theta_S) = \bar{\mathcal{F}}_S(x; \bar{\Theta}_S)^T \mathbf{v}_{\mathcal{I}_{\mathbb{L}}(x')} \tag{1.28}$$

Let N be a context size, and let M be the number of negative samples per true pair. Let $x^{(-)} \sim \mathbb{L}$ be random words (negative samples) drawn from the vocabulary.[1] The parameters Θ_S are trained, via stochastic gradient descent, to minimize the following loss:

$$\mathcal{L}(\Theta_S) = -\sum_{X \in \mathcal{D}_S^{(\text{train})}} \sum_{t \in \{1...|X|\}} \sum_{\substack{t' \in \{1...|X|\}: \\ 0<|t-t'| \leq N}} \left(\ln \sigma(\mathcal{F}_S(x_t, x_{t'}; \Theta_S)) + \sum_{m=1}^{M} \ln \sigma(-\mathcal{F}_S(x_t, x^{(-)}; \Theta_S)) \right) \tag{1.29}$$

Conceptually, the loss function pulls together the $\mathbf{w}$ and $\mathbf{v}$ vectors of word pairs that are truly observed, while pushing apart the $\mathbf{w}$ and $\mathbf{v}$ vectors of random pairs. As a result, words that tend to occur in similar contexts end up with similar $\mathbf{w}$ vectors. For instance, if *bus* often co-occurs with *drive*, and *car* also often co-occurs with *drive*, then *bus* and *car* have similar $\mathbf{w}$ vectors (see Figure 1.5).

1.5.1.2 FastText

Recall that $\mathbb{L}$ is only a finite subset of the infinite set of possible words. One problem with Word2Vec is therefore its inability to assign vectors to unseen words. FastText (Bojanowski et al., 2017) is an extension of Word2Vec that addresses this problem. We use averaged FastText embeddings as part of our sentence meta-embeddings in Chapter 4.

Let $\mathbb{L}' \subset \mathbb{C}^+$ be a limited vocabulary of subwords (character n-grams between some minimum and maximum length), and let $\mathcal{I}_{\mathbb{L}'}$ be the associated indexing function. Let $\mathcal{Z} : \mathbb{C}^+ \rightarrow \mathbb{L}'^+$ be a function that enumerates the subwords of any known or unknown word, e.g., $\mathcal{Z}(bus) = \{\#bus, \#bu, bus\$, us\$, bu, \ldots\}$. Let $\mathbf{W}' \in \mathbb{R}^{|\mathbb{L}'| \times d}$ be an additional parameter matrix in Θ_S and $\bar{\Theta}_S$. Then for transfer learning purposes, the embedding of x is defined as:

$$\bar{\mathcal{F}}_S(x; \bar{\Theta}_S) = \begin{cases} \frac{1}{|\mathcal{Z}(x)|+1} \left(\mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x)} + \sum_{x' \in \mathcal{Z}(x)} \mathbf{w}'_{\mathcal{I}_{\mathbb{L}'}(x')} \right) & \text{if } x \in \mathbb{L} \\ \frac{1}{|\mathcal{Z}(x)|} \sum_{x' \in \mathcal{Z}(x)} \mathbf{w}'_{\mathcal{I}_{\mathbb{L}'}(x')} & \text{otherwise} \end{cases} \tag{1.30}$$

FastText is pretrained in the same way as Word2Vec. The only difference is that we use the compositional word vectors from Equation 1.30 in Equation 1.28.

1.5.1.3 GloVe

GloVe (Global Vectors for Word Representation) (Pennington et al., 2014) is another pretrained model for non-contextualized word embeddings. We use averaged GloVe embeddings as part of our sentence meta-embeddings in Chapter 4.

Like Word2Vec, GloVe is defined over a finite word vocabulary $\mathbb{L}$ with indexing function $\mathcal{I}_{\mathbb{L}}$. Let $\bar{\Theta}_S = \{\mathbf{W}, \mathbf{W}' \in \mathbb{R}^{|\mathbb{L}| \times d}\}$ be two parameter matrices. When using GloVe for transfer learning, Pennington et al. (2014) recommend representing a word $x \in \mathbb{L}$ as:

$$\bar{\mathcal{F}}_S(x; \bar{\Theta}_S) = \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x)} + \mathbf{w}'_{\mathcal{I}_{\mathbb{L}}(x)} \tag{1.31}$$

[1] In practice, the negative samples are not drawn uniformly, but as a function of their frequency in $\mathcal{D}_S^{(\text{train})}$, with undersampling for very frequent words.

During pretraining, GloVe learns to predict the co-occurrence counts of word pairs. Let $\mathcal{C}(x, x')$ be the number of times that $x, x' \in \mathbb{L}$ occurred together within a window of size N in an unsupervised training set $\mathcal{D}_S^{(\text{train})}$. Let $\Theta_S = \bar{\Theta}_S \cup \{\mathbf{b}, \mathbf{b}' \in \mathbb{R}^{|\mathbb{L}|}\}$. GloVe predicts the co-occurrence count as:

$$\mathcal{F}_S(x, x'; \Theta_S) = \exp(\mathbf{w}^T_{\mathcal{I}_\mathbb{L}(x)} \mathbf{w}'_{\mathcal{I}_\mathbb{L}(x')} + b_{\mathcal{I}_\mathbb{L}(x)} + b'_{\mathcal{I}_\mathbb{L}(x')}) \tag{1.32}$$

The loss is the weighted squared error of the true and predicted logarithmic counts:

$$\mathcal{L}(\Theta_S) = \sum_{x \in \mathbb{L}} \sum_{x' \in \mathbb{L}} \mathcal{A}(\mathcal{C}(x, x'))(\ln \mathcal{F}_S(x, x'; \Theta_S) - \ln \mathcal{C}(x, x'))^2 \tag{1.33}$$

where

$$\mathcal{A}(c) = \begin{cases} \left(\frac{c}{100}\right)^{\frac{3}{4}} & \text{if } c < 100 \\ 1 & \text{otherwise} \end{cases} \tag{1.34}$$

1.5.1.4 Wikipedia2Vec

Wikipedia2Vec (Yamada et al., 2016, 2020) is a variant of Word2Vec that learns a joint embedding space of non-contextualized word and entity embeddings. In Chapter 3, we inject pretrained Wikipedia2Vec entity embeddings into the BERT model.

Let $\mathbb{L} = \mathbb{L}^{(\text{word})} \cup \mathbb{L}^{(\text{ent})}$ (with $\mathbb{L}^{(\text{word})} \cap \mathbb{L}^{(\text{ent})} = \emptyset$), be a finite vocabulary of words and entities (Wikipedia pages) with indexing function $\mathcal{I}_\mathbb{L}$. Let $\bar{\Theta}_S = \{\mathbf{W} \in \mathbb{R}^{|\mathbb{L}| \times d}\}$ be the associated parameter matrix. When using Wikipedia2Vec for transfer learning, one would typically initialize the rows of a word or entity embedding lookup layer with:

$$\bar{\mathcal{F}}_S(x; \bar{\Theta}_S) = \mathbf{w}_{\mathcal{I}_\mathbb{L}(x)} \tag{1.35}$$

During pretraining, Wikipedia2Vec predicts whether a given word pair, entity pair or word-entity pair $x, x' \in \mathbb{L}$ is a true co-occurrence in Wikipedia, or a randomly selected pair. Let $\Theta_S = \bar{\Theta}_S \cup \{\mathbf{V} \in \mathbb{R}^{|\mathbb{L}| \times d}\}$, where $\mathbf{V}$ is another parameter matrix. The score of x, x' is defined as:

$$\mathcal{F}_S(x, x'; \Theta_S) = \bar{\mathcal{F}}_S(x, \bar{\Theta}_S)^T \mathbf{v}_{\mathcal{I}_\mathbb{L}(x')} \tag{1.36}$$

Wikipedia2Vec has three source tasks with losses $\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \mathcal{L}^{(3)}$. The first task is standard skip-gram with negative sampling, i.e., $\mathcal{L}^{(1)}$ is Equation 1.29.

The second task is to classify which entity pairs are neighbors in the Wikipedia link graph. Let $\mathcal{N}(c)$ be the set of neighbors of entity c, and let M be the number of negative samples per positive pair. Let $c^{(-)} \sim \mathbb{L}^{(\text{ent})}$ be randomly selected entities (negative samples). Then the loss is:

$$\mathcal{L}^{(2)}(\Theta_S) = -\sum_{c \in \mathbb{L}^{(\text{ent})}} \sum_{c^{(+)} \in \mathcal{N}(c)} \left(\ln \sigma(\mathcal{F}_S(c, c^{(+)}; \Theta_S)) + \sum_{m=1}^{M} \ln \sigma(-\mathcal{F}_S(c, c^{(-)}; \Theta_S)) \right) \tag{1.37}$$

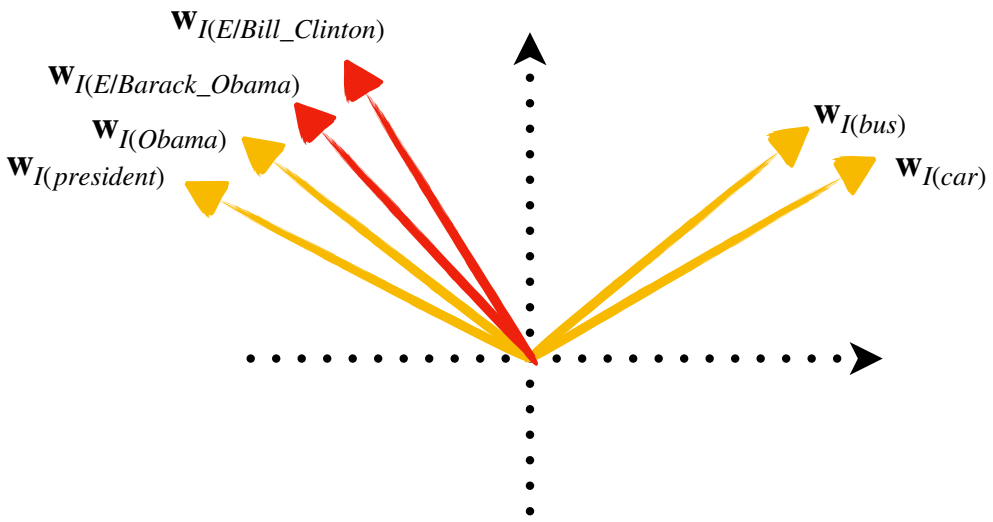

Figure 1.6: Schematic depiction of pretrained Wikipedia2Vec vectors: Words that are semantically similar get similar word vectors (yellow and yellow vectors). Entities that are semantically similar get similar entity vectors (red and red vectors). Entities and their names get similar word and entity vectors (yellow and red vectors).

The third task is a variant of skip-gram with negative sampling, where the model predicts which words occur with which entity hyperlinks (and vice versa). Let $\mathcal{H}(X) = \{(t_1, n_1, c_1) \dots (t_H, n_H, c_H)\}$ be a set of start positions, lengths and entities pertaining to sentence X, such that $X_{t:t+n}$ is a hyperlink towards the Wikipedia page of $c \in \mathbb{L}^{(\text{ent})}$. Let N be a context size, and let $x^{(-)} \sim \mathbb{L}^{(\text{word})}$ be randomly selected words, as described in Section 1.5.1.1. Then the third loss is:

$$\mathcal{L}^{(3)}(\Theta_S) = -\sum_{X \in \mathcal{D}_S^{(\text{train})}} \sum_{\substack{(t,n,c) \in \\ \mathcal{H}(X)}} \sum_{\substack{t' \in \{1 \dots |X|\}: \\ (t-N \leq t' < t) \vee \\ (t+n < t' \leq t+n+N)}} \Big(\ln \sigma(\mathcal{F}_S(c, x_{t'}; \Theta_S)) + \ln \sigma(\mathcal{F}_S(x_{t'}, c; \Theta_S)) + \sum_{m=1}^{M} \left(\ln \sigma(-\mathcal{F}_S(x_{t'}, c^{(-)}; \Theta_S)) + \ln \sigma(-\mathcal{F}_S(c, x^{(-)}; \Theta_S)) \right) \Big) \tag{1.38}$$

The joint loss is $\mathcal{L}(\Theta_S) = \mathcal{L}^{(1)}(\Theta_S) + \mathcal{L}^{(2)}(\Theta_S) + \mathcal{L}^{(3)}(\Theta_S)$. Minimizing this loss yields a vector space where:

- semantically similar words have similar word vectors (e.g., *car* and *bus*)
- semantically similar entities have similar entity vectors (e.g., *E/Barack_Obama* and *E/Bill_Clinton*)
- entities and the words that are used to refer to them have similar entity and word vectors (e.g., *E/Barack_Obama* and *Obama*)

See Figure 1.6 for a schematic depiction of a Wikipedia2Vec embedding space.

1.5.1.5 ParaNMT

ParaNMT (Wieting and Gimpel, 2018) is a pretrained model that produces sentence embeddings for sentence similarity tasks. Here, we describe a variant of ParaNMT that is based on averaged non-contextualized word and 3-gram embeddings. We use it as part of our sentence meta-embeddings in Chapter 5.

Let $\mathbb{L}$ be a vocabulary of words with indexing function $\mathcal{I}_{\mathbb{L}}$, and let $\mathbb{L}'$ be a vocabulary of character 3-grams with indexing function $\mathcal{I}_{\mathbb{L}'}$. Let $\Theta_S = \bar{\Theta}_S = \{\mathbf{W} \in \mathbb{R}^{|\mathbb{L}| \times d}, \mathbf{W}' \in \mathbb{R}^{|\mathbb{L}'| \times d'}\}$ be the associated parameter matrices. Let $X = x_1 \ldots x_T$ be a word-tokenized sentence, and let $\mathcal{Z} : \mathbb{C}^+ \rightarrow \mathbb{L}'^+$ be a function that enumerates the character 3-grams of a given word, similar to the function described in Section 1.5.1.2. Then, ParaNMT embeds X as:

$$\bar{\mathcal{F}}_S(X; \bar{\Theta}_S) = \Big[\frac{1}{|X|}\sum_{t=1}^{|X|} \mathbf{w}_{\mathcal{I}_{\mathbb{L}}(x_t)} \quad ; \quad \frac{1}{\sum_{t=1}^{|X|} |\mathcal{Z}(x_t)|}\sum_{t=1}^{|X|}\sum_{x' \in \mathcal{Z}(x_t)} \mathbf{w}'_{\mathcal{I}_{\mathbb{L}'}(x')}\Big] \tag{1.39}$$

During pretraining, ParaNMT learns to distinguish true synthetic paraphrases from random paraphrases. To generate synthetic paraphrases, Wieting and Gimpel (2018) translate English sentences into another language and back, using a separate Neural Machine Translation system. ParaNMT is therefore a weakly supervised pretrained model.

Let $\mathcal{D}_S^{(\text{train})}$ be a corpus of sentences X and their true paraphrases $X^{(+)}$. Let $X^{(-)}$ be random paraphrases, e.g., the paraphrases of other, randomly drawn sentences. Then, the loss function is the max margin loss of the cosine similarities, with margin δ:

$$\begin{aligned}\mathcal{L}(\Theta_S) = \sum_{(X, X^{(+)}) \in \mathcal{D}_S^{(\text{train})}} \max\Big(0, \delta - \text{sim}(\bar{\mathcal{F}}_S(X; \bar{\Theta}_S), \bar{\mathcal{F}}_S(X^{(+)}; \bar{\Theta}_S)) \\ + \text{sim}(\bar{\mathcal{F}}_S(X; \bar{\Theta}_S), \bar{\mathcal{F}}_S(X^{(-)}; \bar{\Theta}_S))\Big)\end{aligned} \tag{1.40}$$

where

$$\text{sim}(\mathbf{s}, \mathbf{s}') = \frac{\mathbf{s}^T \mathbf{s}'}{||\mathbf{s}||_2 ||\mathbf{s}'||_2} \tag{1.41}$$

1.5.2 Pretrained models for contextualized embeddings

1.5.2.1 ELMo

ELMo (Embeddings from Language Models) (Peters et al., 2018) was one of the first pretrained contextualized embedding models for NLP. In Chapter 4, we use ELMo as a baseline for duplicate question detection.

ELMo is a multi-layer BiLSTM. Let $X = x_1 \ldots x_T$ be a sequence of words from vocabulary $\mathbb{L}$. Every x_t is further tokenized into a sequence of characters, which are encoded into a non-contextualized word embedding $\mathbf{e}_t$ by a character CNN with parameters $\Theta^{(\text{cnn})}$.

The input to the BiLSTM is defined as:

$$\overleftarrow{\mathbf{H}}^{(0)} = \overrightarrow{\mathbf{H}}^{(0)} = \begin{bmatrix} — & \mathbf{e}_1 & — \\ & \vdots & \\ — & \mathbf{e}_T & — \end{bmatrix}$$

The input is fed into two stacks of multi-layer left-to-right and right-to-left LSTMs, whose parameters are denoted $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$. The output are two sets of contextualized word embeddings, $\{\overrightarrow{\mathbf{H}}^{(1)} \dots \overrightarrow{\mathbf{H}}^{(L)}\}$ and $\{\overleftarrow{\mathbf{H}}^{(1)} \dots \overleftarrow{\mathbf{H}}^{(L)}\}$, with $\mathbf{H}^{(*)} \in \mathbb{R}^{|X| \times d}$. When transferring ELMo to a downstream task, the contextualized embeddings are usually combined via a weighted sum, with trainable scalar parameters:

$$\bar{\mathcal{F}}_S(X; \bar{\Theta}_S) = \gamma \sum_{l=0}^{L} \lambda_l [\overrightarrow{\mathbf{H}}^{(l)}; \overleftarrow{\mathbf{H}}^{(l)}] \tag{1.42}$$

The transferred parameters are thus:

$$\bar{\Theta}_S = \{\gamma, \lambda_0 \dots \lambda_L\} \cup \Theta^{(\text{cnn})} \cup \overrightarrow{\Theta} \cup \overleftarrow{\Theta} \tag{1.43}$$

During pretraining, the LSTMs serve as left-to-right and right-to-left autoregressive language models. Let $\mathcal{F}^{(\text{lm})} : \mathbb{R}^d \to \mathbb{R}^{|\mathbb{L}|}$ be an additional linear layer with parameters $\Theta^{(\text{lm})}$. Let $\Theta_S = \Theta^{(\text{cnn})} \cup \overrightarrow{\Theta} \cup \overleftarrow{\Theta} \cup \Theta^{(\text{lm})}$. Then the autoregressive language models are defined as:

$$\begin{aligned} \overrightarrow{\hat{P}}(y|X_{1:t-1}; \Theta_S) &\propto \exp(\mathcal{F}^{(\text{lm})}(\overrightarrow{\mathbf{h}}_{t-1}; \Theta^{(\text{lm})})_{\mathcal{I}_{\mathbb{L}}(y)}) \\ \overleftarrow{\hat{P}}(y|X_{t+1:|X|}; \Theta_S) &\propto \exp(\mathcal{F}^{(\text{lm})}(\overleftarrow{\mathbf{h}}_{t+1}; \Theta^{(\text{lm})})_{\mathcal{I}_{\mathbb{L}}(y)}) \end{aligned} \tag{1.44}$$

ELMo is trained to minimize the negative log likelihood of the tokens of the source training set:

$$\mathcal{L}(\Theta_S) = -\sum_{X \in \mathcal{D}_S^{(\text{train})}} \frac{1}{|X|} \sum_{t=1}^{|X|} \left(\ln \overrightarrow{\hat{P}}(x_t|X_{1:t-1}; \Theta_S) + \ln \overleftarrow{\hat{P}}(x_t|X_{t+1:|X|}; \Theta_S) \right) \tag{1.45}$$

While ELMo is a bidirectional LSTM, its bidirectionality is "shallow", since its forward and backward LSTMs do not communicate with each other. If they did, the forward LSTM would receive information about words to the right of the current position (and vice versa), which would render the language modeling task trivial.

1.5.2.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) was proposed as a solution to the lack of deep bidirectionality in autoregressive language

models. It has since become one of the most popular pretrained models in the NLP community (Wolf et al., 2020). We use BERT in Chapters 2, 3 and 4.

BERT is a pretrained Transformer model. One of its innovative features is the ability to encode single texts X and text pairs $X^{(1)}, X^{(2)}$ with the same architecture. Single texts are tokenized as:

$$X = \textit{[CLS]}, x_1 \dots x_T, \textit{[SEP]} \tag{1.46}$$

where *[SEP]*, *[CLS]* $\in \mathbb{L}$ are special tokens. Text pairs are tokenized as:

$$X = \textit{[CLS]}, x_1^{(1)} \dots x_{T_1}^{(1)}, \textit{[SEP]}, x_1^{(2)} \dots x_{T_2}^{(2)}, \textit{[SEP]} \tag{1.47}$$

To differentiate between text pairs, BERT adds special trainable embeddings $\mathbf{g}^{(1)}, \mathbf{g}^{(2)} \in \mathbb{R}^d$ to Equation 1.25. For simplicity, we will abstract away from the special tokens and the single text/text pair distinction, and refer to any input as $X = x_1 \dots x_T$.

BERT produces contextualized token embeddings $\mathbf{H} \in \mathbb{R}^{|X| \times d}$, where $\mathbf{H} = \mathcal{F}^{(\text{tf})}(X; \Theta^{(\text{tf})})$ (see Section 1.4.3.2). It can also produce one sentence embedding $\mathbf{s} \in \mathbb{R}^d$ that summarizes the entire input. Let $\mathcal{F}^{(\text{pool})}$ be an additional linear layer with parameters $\Theta^{(\text{pool})}$. Then:

$$\mathbf{s} = \tanh(\mathcal{F}^{(\text{pool})}(\mathbf{h}_1; \Theta^{(\text{pool})})) \tag{1.48}$$

When using BERT for transfer learning, one would either define $\bar{\mathcal{F}}_S(X; \bar{\Theta}_S) = \mathbf{H}$ or $\bar{\mathcal{F}}_S(X; \bar{\Theta}_S) = \mathbf{s}$, depending on the type of embedding that is required. The transferred parameters are thus $\bar{\Theta}_S = \Theta^{(\text{tf})}$ or $\bar{\Theta}_S = \Theta^{(\text{tf})} \cup \Theta^{(\text{pool})}$.

BERT is pretrained on two source tasks: masked language modeling (MLM) and next sentence prediction (NSP). Let $\Theta^{(\text{mlm})}$ and $\Theta^{(\text{nsp})}$ be the parameters associated with each task. The full set of source model parameters is thus:

$$\Theta_S = \bar{\Theta}_S \cup \Theta^{(\text{mlm})} \cup \Theta^{(\text{nsp})} \tag{1.49}$$

NSP: NSP is a binary classification task. Given two texts $X^{(1)}, X^{(2)}$, which are concatenated into a single input $X = x_1 \dots x_T$ as described above, the model is trained to predict whether $X^{(1)}$ and $X^{(2)}$ are consecutive sequences of the source training set $\mathcal{D}_S^{(\text{train})}$, or whether they are randomly combined. (In practice, the true input to NSP is a noisy version of X, denoted $\bar{X}$, which will be described below.)

The classification is done by a linear layer $\mathcal{F}^{(\text{nsp})} : \mathbb{R}^d \to \mathbb{R}^2$, which sits on top of the sentence embedding $\mathbf{s}$. The output space of the NSP task is $\mathcal{Y} = \{\textit{C(onsecutive)}, \textit{R(andom)}\}$. The probability of $Y \in \mathcal{Y}$ is predicted as:

$$\hat{P}^{(\text{nsp})}(Y|\bar{X}; \Theta_S) \propto \exp(\mathcal{F}^{(\text{nsp})}(\mathbf{s}; \Theta^{(\text{nsp})})_{\mathcal{I}_{\mathcal{Y}}(Y)}) \tag{1.50}$$

The loss is the negative log likelihood of the true labels, which are created during the preprocessing of the training set:

$$\mathcal{L}^{(\text{nsp})}(\Theta_S) = - \sum_{(X,Y) \in \mathcal{D}_S^{(\text{train})}} \ln \hat{P}^{(\text{nsp})}(Y|\bar{X}; \Theta_S) \tag{1.51}$$

MLM: The MLM task defines the noisy input $\bar{X}$ by selecting up to 15% of available token positions in X for masking. Let $\mathcal{Q}(X) \subset \{1 \dots |X|\}$ be the set of selected positions, and let $r \sim \mathcal{U}(0,1)$ be a random uniform variable. Let *[MASK]* $\in \mathbb{L}$ be a special token. Then $\bar{X}$ is constructed as follows:

$$\bar{x}_t = \begin{cases} \textit{[MASK]} & \text{if } t \in \mathcal{Q}(X) \wedge r \leq 0.8 \\ x' \sim \mathbb{L} & \text{if } t \in \mathcal{Q}(X) \wedge 0.8 < r \leq 0.9 \\ x_t & \text{otherwise} \end{cases} \tag{1.52}$$

The model tries to reconstruct the original tokens. To do so, it predicts probability distributions over $\mathbb{L}$ via a feed forward layer $\mathcal{F}^{(\text{mlm})}$, which sits on top of the contextualized token embeddings:

$$\hat{P}_t^{(\text{mlm})}(y|\bar{X};\Theta_S) \propto \exp(\mathcal{F}_S^{(\text{mlm})}(\mathbf{h}_t;\Theta^{(\text{mlm})})_{\mathcal{I}_{\mathbb{L}}(y)}) \tag{1.53}$$

The MLM loss is the average negative log likelihood of the original tokens:

$$\mathcal{L}^{(\text{mlm})}(\Theta_S) = -\sum_{(X,Y)\in\mathcal{D}_S^{(\text{train})}} \frac{1}{|\mathcal{Q}(X)|} \sum_{t\in\mathcal{Q}(X)} \ln \hat{P}_t^{(\text{mlm})}(x_t|\bar{X};\Theta_S) \tag{1.54}$$

MLM is not an autoregressive objective. This means that if two words $x_t, x_{t'}$ are masked at the same time, the model cannot learn $\hat{P}_t(x_t|x_{t'})$ or $\hat{P}_{t'}(x_{t'}|x_t)$. The more recently proposed XLNet model (Yang et al., 2019) addresses this problem by factorizing the word order. Empirical studies by Liu et al. (2019b) suggest that the non-autoregressive nature of MLM is not a big problem in practice, especially if masking patterns vary between training epochs.

1.5.2.3 InferSent

InferSent (Conneau et al., 2017) is a sentence encoder for sentence-level transfer learning. It produces one embedding $\mathbf{s} \in \mathbb{R}^d$ per sentence X. In Chapter 4, we use InferSent as a baseline for duplicate question detection.

InferSent is a BiLSTM on top of non-contextualized word vectors. Let $\bar{\Theta}_S$ be the combined parameters of the BiLSTM and word vectors, and let $\mathbf{H}$ be the matrix of contextualized word embeddings returned by the BiLSTM, as described in Section 1.4.3.1. The sentence embedding is derived by global max pooling:

$$\bar{\mathcal{F}}_S(X;\bar{\Theta}_S) = \mathbf{s} = \begin{bmatrix} \max_t(h_{t,1}) \\ \vdots \\ \max_t(h_{t,d}) \end{bmatrix} \tag{1.55}$$

The non-contextualized embeddings are initialized with pretrained GloVe embeddings, i.e., some parameters of InferSent were implicitly pretrained on the GloVe task (see Section 1.5.1.3). Additionally, the BiLSTM is pretrained on the Stanford natural language

inference (SNLI) dataset (Bowman et al., 2015), which is a popular dataset for sentence-level transfer learning. SNLI is a sentence-pair classification task, where a model predicts what relationship holds between premise and hypothesis sentences. The input space is the space of possible sentence pairs $\mathcal{X} = \mathbb{L}^+ \times \mathbb{L}^+$, and the output space is $\mathcal{Y} = \{E(ntails), C(ontradicts), N(eutral)\}$.

Let $\mathcal{F}^{(\text{nli})}$ be a feed forward layer with parameters $\Theta^{(\text{nli})} = \{\mathbf{W}^{(1)} \in \mathbb{R}^{d' \times 4d}, \mathbf{W}^{(2)} \in \mathbb{R}^{|\mathcal{Y}| \times d'}, \mathbf{b}^{(1)} \in \mathbb{R}^{d'}, \mathbf{b}^{(2)} \in \mathbb{R}^{|\mathcal{Y}|}\}$ and an elementwise non-linearity $\mathcal{G}$. $\mathcal{F}^{(\text{nli})}$ operates on the concatenated sentence embeddings, their elementwise absolute difference and Hadamard product:

$$\mathcal{F}^{(\text{nli})}(\mathbf{s}^{(1)}, \mathbf{s}^{(2)}; \Theta^{(\text{nli})}) = \mathbf{W}^{(2)}\mathcal{G}(\mathbf{W}^{(1)}[\mathbf{s}^{(1)}; \mathbf{s}^{(2)}; \text{abs}(\mathbf{s}^{(1)} - \mathbf{s}^{(2)}); \mathbf{s}^{(1)} \odot \mathbf{s}^{(2)}] + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} \quad (1.56)$$

Let $\Theta_S = \bar{\Theta}_S \cup \Theta^{(\text{nli})}$. The probability of the label $Y \in \mathcal{Y}$, given a premise-hypothesis pair, is defined as:

$$\hat{P}(Y|X^{(1)}, X^{(2)}; \Theta_S) \propto \exp\Big(\mathcal{F}^{(\text{nli})}(\bar{\mathcal{F}}_S(X^{(1)}; \bar{\Theta}_S), \bar{\mathcal{F}}_S(X^{(2)}; \bar{\Theta}_S); \Theta^{(\text{nli})})_{\mathcal{I}_{\mathcal{Y}}(Y)}\Big) \quad (1.57)$$

The loss is then simply the negative log likelihood of the true labels in the SNLI training set:

$$\mathcal{L}(\Theta_S) = - \sum_{((X^{(1)}, X^{(2)}), Y) \in \mathcal{D}_S^{(\text{train})}} \ln \hat{P}(Y|X^{(1)}, X^{(2)}; \Theta_S) \quad (1.58)$$

1.5.2.4 USE

USE (Universal Sentence Encoder) (Cer et al., 2018) is a more recent pretrained sentence encoder, which is based on the Transformer architecture. In contrast to InferSent and the other pretrained models described in this section, the source code and some of the training data of USE are not publicly available. As a result, it is difficult to reproduce or customize the pretraining process, and the model is normally used as an "off-the-shelf" blackbox. We use USE as part of our sentence meta-embeddings in Chapters 4 and 5.

Like InferSent, USE produces one sentence embedding $\mathbf{s} \in \mathbb{R}^d$ per input X. The model is pretrained on the following source tasks:

- Skip-thought (Kiros et al., 2015), which is similar to the NSP task.
- Response prediction (Henderson et al., 2017; Yang et al., 2018): Selecting the correct next turn in conversational data.
- SNLI (Bowman et al., 2015), see Section 1.5.2.3.

1.5.2.5 SBERT

Without additional supervision, sentence embeddings from BERT are not optimally suited for sentence similarity tasks. SBERT (SentenceBERT) (Reimers and Gurevych, 2019) is a modification of BERT that addresses this problem. We use SBERT as part of our sentence meta-embeddings in Chapter 5.

SBERT embeds sentences by taking the unweighted average over the contextualized token embeddings returned by the BERT architecture:

$$\bar{\mathcal{F}}_S(X; \bar{\Theta}_S) = \frac{1}{|X|} \sum_{t=1}^{|X|} \mathbf{h}_t \tag{1.59}$$

where $\mathbf{h}_t$ and $\bar{\Theta}_S = \Theta^{(\text{tf})}$ are defined in Section 1.5.2.2.

Due to the initialization with BERT, SBERT is implicitly pretrained on the MLM and NSP tasks described in Section 1.5.2.2. Additionally, Reimers and Gurevych (2019) pretrain SBERT on SNLI (Bowman et al., 2015), using an architecture that is similar to the one described in Section 1.5.2.3. There are versions of SBERT that are pretrained on other sentence pair tasks as well, however, we use the SNLI-only version in Chapter 5.

1.6 Summary of introduction

In Section 1.1.1, we stated the motivation of this thesis: Contextualized embeddings are powerful, but they are expensive to pretrain and adapt. A worthwhile alternative is to enhance existing contextualized embedding models with cheaper non-contextualized embeddings.

In Section 1.1.2, we stated our contributions to this line of research: injecting non-contextualized word or entity embeddings into the BERT model (Chapters 2, 3), and creating meta-embeddings from contextualized and non-contextualized sentence embeddings (Chapters 4, 5).

In Sections 1.2 through 1.5, we provided some necessary background information: We introduced core concepts of deep learning and transfer learning, and we described the architectures and pretrained models that are used in this thesis.

The remainder of this thesis consists of the four publications that were outlined in Section 1.1.2.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, USA.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, USA.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Francis R Bach and Michael I Jordan. 2002. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48.

Iz Beltagy, Kyle Lo, and Arman Cohan. 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*, pages 3606–3611, Hong Kong, China.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics*, pages 177–186, Paris, France. Springer.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 1–14, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 169–174, Brussels, Belgium.

Kenneth Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pages 1699–1704, Miyazaki, Japan.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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*, pages 4171–4186, Minneapolis, USA.

Philipp Dufter and Hinrich Schütze. 2019. Analytical methods for interpretable ultradense word embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1185–1191, Hong Kong, China.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Proceedings of the Third Workshop on Representation Learning for NLP*, pages 91–100, Melbourne, Australia.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, USA.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 34–45, Melbourne, Australia.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In *Australasian Document Computing Symposium*, Parramatta, Australia.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 328–339, Melbourne, Australia.

Jon R Kettenring. 1971. Canonical analysis of several sets of variables. *Biometrika*, 58(3):433–451.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302, Montreal, Canada.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 66–71, Brussels, Belgium.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, USA.

Preslav Nakov, Doris Hoogeveen, Lluis Marquez, Allessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community question answering. In *International Workshop on Semantic Evaluation*, pages 27–48, Vancouver, Canada.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65, Florence, Italy.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543, Doha, Qatar.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1765, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, USA.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 556–566, Denver, USA.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2020. Snorkel: rapid training data creation with weak supervision. *The International Journal on Very Large Data Bases*, 29(2):709–730.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, San Diego, USA.

Sebastian Ruder. 2019. *Neural transfer learning for natural language processing*. Ph.D. thesis, NUI Galway.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Hinrich Schütze. 1992. Dimensions of meaning. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, pages 787–796, Minneapolis, USA.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations*, Toulon, France.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 13693–13696, New York, USA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, USA.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 451–462, Melbourne, Australia.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 38–45, Online.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, USA.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 23–30, Online.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, Berlin, Germany.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. In *Proceedings of the Third Workshop of Representation Learning for NLP*, pages 164–174, Melbourne, Australia.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, Vancouver, Canada.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1351–1360, Berlin, Germany.

Chapter 2

Corresponds to the following publication:

> Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and Covid-19 QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (presented at SustaiNLP: Workshop on Simple and Efficient Natural Language Processing)*, pages 1482–1490, Online

Declaration of Co-Authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission. Ulli Waltinger contributed by reviewing an earlier draft of the paper.

[1] `https://www.aclweb.org/anthology/faq/copyright/` [accessed 05/03/2021]

Inexpensive Domain Adaptation of Pretrained Language Models: Case Studies on Biomedical NER and Covid-19 QA

Nina Poerner[*†] **and Ulli Waltinger**[†] **and Hinrich Schütze**[*]
[*]Center for Information and Language Processing, LMU Munich, Germany
[†]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

Abstract

Domain adaptation of Pretrained Language Models (PTLMs) is typically achieved by unsupervised pretraining on target-domain text. While successful, this approach is expensive in terms of hardware, runtime and CO_2 emissions. Here, we propose a cheaper alternative: We train Word2Vec on target-domain text and align the resulting word vectors with the wordpiece vectors of a general-domain PTLM. We evaluate on eight English biomedical Named Entity Recognition (NER) tasks and compare against the recently proposed BioBERT model. We cover over 60% of the BioBERT – BERT F1 delta, at 5% of BioBERT's CO_2 footprint and 2% of its cloud compute cost. We also show how to quickly adapt an existing general-domain Question Answering (QA) model to an emerging domain: the Covid-19 pandemic.[1]

1 Introduction

Pretrained Language Models (PTLMs) such as BERT (Devlin et al., 2019) have spearheaded advances on many NLP tasks. Usually, PTLMs are pretrained on unlabeled general-domain and/or mixed-domain text, such as Wikipedia, digital books or the Common Crawl corpus.

When applying PTLMs to specific domains, it can be useful to domain-adapt them. Domain adaptation of PTLMs has typically been achieved by pretraining on target-domain text. One such model is BioBERT (Lee et al., 2020), which was initialized from general-domain BERT and then pretrained on biomedical scientific publications. The domain adaptation is shown to be helpful for target-domain tasks such as biomedical Named Entity Recognition (NER) or Question Answering (QA). On the downside, the computational cost of pretraining can be considerable: BioBERTv1.0 was adapted for ten days on eight large GPUs (see Table 1), which is expensive, environmentally unfriendly, prohibitive for small research labs and students, and may delay prototyping on emerging domains.

We therefore propose a **fast, CPU-only domain-adaptation method for PTLMs**: We train Word2Vec (Mikolov et al., 2013a) on target-domain text and align the resulting word vectors with the wordpiece vectors of an existing general-domain PTLM. The PTLM thus gains domain-specific lexical knowledge in the form of additional word vectors, but its deeper layers remain unchanged. Since Word2Vec and the vector space alignment are efficient models, the process requires a fraction of the resources associated with pretraining the PTLM itself, and it can be done on CPU.

In Section 4, we use the proposed method to domain-adapt BERT on PubMed+PMC (the data used for BioBERTv1.0) and/or CORD-19 (Covid-19 Open Research Dataset). We improve over general-domain BERT on eight out of eight biomedical NER tasks, using a fraction of the compute cost associated with BioBERT. In Section 5, we show how to quickly adapt an existing Question Answering model to text about the Covid-19 pandemic, without any target-domain Language Model pretraining or finetuning.

2 Related work

2.1 The BERT PTLM

For our purpose, a PTLM consists of three parts: A tokenizer $\mathcal{T}_{\text{LM}} : \mathbb{L}^+ \rightarrow \mathbb{L}^+_{\text{LM}}$, a wordpiece embedding lookup function $\mathcal{E}_{\text{LM}} : \mathbb{L}_{\text{LM}} \rightarrow \mathbb{R}^{d_{\text{LM}}}$ and an encoder function $\mathcal{F}_{\text{LM}}$. $\mathbb{L}_{\text{LM}}$ is a limited vocabulary of wordpieces. All words from the natural language $\mathbb{L}^+$ that are not in $\mathbb{L}_{\text{LM}}$ are tokenized into sequences of shorter wordpieces, e.g., *dementia* becomes *dem ##ent ##ia*. Given a sentence $S = [w_1, \ldots, w_T]$, tokenized

[1] www.github.com/npoe/covid-qa

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1482–1490

	size	Domain adaptation hardware	Power(W)	Time(h)	CO_2(lbs)	Google Cloud $
BioBERTv1.0	base	8 NVIDIA v100 GPUs (32GB)	1505	240	544	1421 – 4762
BioBERTv1.1	base	8 NVIDIA v100 GPUs (32GB)	1505	552	1252	3268 – 10952
GreenBioBERT (Section 4)	base	12 Intel Xeon E7-8857 CPUs, 30GB RAM	1560	12	28	16 – 76
GreenCovidSQuADBERT (Section 5)	large	12 Intel Xeon E7-8857 CPUs, 40GB RAM	1560	24	56	32 – 152

Table 1: Domain adaptation cost. CO_2 emissions are calculated according to Strubell et al. (2019). Since our hardware configuration is not available on Google Cloud, we take an *m1-ultramem-40* instance (40 vCPUs, 961GB RAM) to estimate an upper bound on our Google Cloud cost.

as $\mathcal{T}_{\text{LM}}(S) = [\mathcal{T}_{\text{LM}}(w_1); \ldots; \mathcal{T}_{\text{LM}}(w_T)]$, $\mathcal{E}_{\text{LM}}$ embeds every wordpiece in $\mathcal{T}_{\text{LM}}(S)$ into a real-valued, trainable wordpiece vector. The wordpiece vectors of the entire sequence are stacked and fed into $\mathcal{F}_{\text{LM}}$. Note that we consider position and segment embeddings to be a part of $\mathcal{F}_{\text{LM}}$ rather than $\mathcal{E}_{\text{LM}}$.

In the case of BERT, $\mathcal{F}_{\text{LM}}$ is a Transformer (Vaswani et al., 2017), followed by a final Feed-Forward Net. During pretraining, the Feed-Forward Net predicts the identity of masked wordpieces. When finetuning on a supervised task, it is usually replaced with a randomly initialized layer.

2.2 Domain-adapted PTLMs

Domain adaptation of PTLMs is typically achieved by pretraining on unlabeled target-domain text. Some examples of such models are BioBERT (Lee et al., 2020), which was pretrained on the PubMed and/or PubMed Central (PMC) corpora, SciBERT (Beltagy et al., 2019), which was pretrained on papers from SemanticScholar, ClinicalBERT (Alsentzer et al., 2019; Huang et al., 2019a) and ClinicalXLNet (Huang et al., 2019b), which were pretrained on clinical patient notes, and AdaptaBERT (Han and Eisenstein, 2019), which was pretrained on Early Modern English text. In most cases, a domain-adapted PTLM is initialized from a general-domain PTLM (e.g., standard BERT), though Beltagy et al. (2019) report better results with a model that was pretrained from scratch with a custom wordpiece vocabulary. In this paper, we focus on BioBERT, as its domain adaptation corpora are publicly available.

	Acc@1	Acc@5	Acc@10
train (19.8K words)	53.6	63.5	65.7
heldout (2.2K words)	39.4	51.6	54.3

Table 2: $\mathbb{L}_{\text{W2V}} \rightarrow \mathbb{L}_{\text{LM}}$ alignment accuracy (%), i.e., how often the identical string is in the top-K nearest neighbors.

2.3 Word vectors

Word vectors are distributed representations of words that are trained on unlabeled text. Contrary to PTLMs, word vectors are non-contextual, i.e., a word type is always assigned the same vector, regardless of context. In this paper, we use Word2Vec (Mikolov et al., 2013a) to train word vectors. We will denote the Word2Vec lookup function as $\mathcal{E}_{\text{W2V}} : \mathbb{L}_{\text{W2V}} \rightarrow \mathbb{R}^{d_{\text{W2V}}}$.

2.4 Word vector space alignment

Word vector space alignment has most frequently been explored in the context of cross-lingual word embeddings. For instance, Mikolov et al. (2013b) align English and Spanish Word2Vec spaces by a simple linear transformation. Wang et al. (2019) use a related method to align cross-lingual word vectors and multilingual BERT wordpiece vectors. In this paper, we apply the method to the problem of domain adaptation within the same language.

3 Method

In the following, we assume access to a general-domain PTLM, as described in Section 2.1, and a corpus of unlabeled target-domain text.

3.1 Creating new input vectors

In a first step, we train Word2Vec on the target-domain corpus. In a second step, we take the intersection of $\mathbb{L}_{\text{LM}}$ and $\mathbb{L}_{\text{W2V}}$. In practice, the intersection mostly contains wordpieces from $\mathbb{L}_{\text{LM}}$ that correspond to standalone words. It also contains single characters and other noise, however, we found that filtering them does not improve alignment quality. In a third step, we use the intersection to fit an unconstrained linear transformation $\mathbf{W} \in \mathbb{R}^{d_{\text{LM}} \times d_{\text{W2V}}}$ via least squares:

$$\underset{\mathbf{W}}{\text{argmin}} \sum_{x \in \mathbb{L}_{\text{LM}} \cap \mathbb{L}_{\text{W2V}}} ||\mathbf{W}\mathcal{E}_{\text{W2V}}(x) - \mathcal{E}_{\text{LM}}(x)||_2^2$$

Intuitively, $\mathbf{W}$ makes Word2Vec vectors "look like" the PTLM's native wordpiece vectors, just

	Query	NNs of query in $\mathcal{E}_{\text{LM}}[\mathbb{L}_{\text{LM}}]$	NNs of query in $\mathbf{W}\mathcal{E}_{\text{W2V}}[\mathbb{L}_{\text{W2V}}]$
query $\in \mathbb{L}_{\text{W2V}} \cap \mathbb{L}_{\text{LM}}$ **Boldface:** Training vector pairs	**surgeon**	physician, psychiatrist, surgery	**surgeon**, urologist, neurosurgeon
	surgeon	**surgeon**, physician, researcher	neurosurgeon, urologist, radiologist
	depression	Depression, recession, depressed	**depression**, Depression, hopelessness
	depression	**depression**, anxiety, anxiousness	depressive, insomnia, Depression
	fatal	lethal, deadly, disastrous	**fatal**, lethal, deadly
	fatal	**fatal**, catastrophic, disastrous	lethal, devastating, disastrous
query $\in \mathbb{L}_{\text{W2V}} - \mathbb{L}_{\text{LM}}$	ventricular	cardiac, pulmonary, mitochondrial	atrial, ventricle, RV
	dementia	diabetes, Alzheimer, autism	VaD, MCI, AD
	suppressants	medications, medicines, medication	suppressant, prokinetics, painkillers
	anesthesiologist	surgeon, technician, psychiatrist	anesthetist, anaesthesiologist, anaesthetist
	nephrotoxicity	toxicity, inflammation, contamination	hepatotoxicity, ototoxicity, cardiotoxicity
	impairment	inability, disruption, disorders	impairments, deficits, deterioration

Table 3: Examples of within-space and cross-space nearest neighbors (NNs) by cosine similarity in GreenBioBERT's wordpiece embedding layer. Blue: Original wordpiece space. Green: Aligned Word2Vec space.

like cross-lingual alignment makes word vectors from one language "look like" word vectors from another language. In Table 2, we report word alignment accuracy when we split $\mathbb{L}_{\text{LM}} \cap \mathbb{L}_{\text{W2V}}$ into a training and development set.[2] In Table 3, we show examples of within-space and cross-space nearest neighbors after alignment.

3.2 Updating the wordpiece embedding layer

Next, we redefine the wordpiece embedding layer of the PTLM. The most radical strategy would be to replace the entire layer with the aligned Word2Vec vectors:

$$\hat{\mathcal{E}}_{\text{LM}} : \mathbb{L}_{\text{W2V}} \rightarrow \mathbb{R}^{d_{\text{LM}}} \; ; \; \hat{\mathcal{E}}_{\text{LM}}(x) = \mathbf{W}\mathcal{E}_{\text{W2V}}(x)$$

In initial experiments, this strategy led to a drop in performance, presumably because function words are not well represented by Word2Vec, and replacing them disrupts BERT's syntactic abilities. To prevent this problem, we leave existing wordpiece vectors intact and only add new ones:

$$\hat{\mathcal{E}}_{\text{LM}} : \mathbb{L}_{\text{LM}} \cup \mathbb{L}_{\text{W2V}} \rightarrow \mathbb{R}^{d_{\text{LM}}};$$
$$\hat{\mathcal{E}}_{\text{LM}}(x) = \begin{cases} \mathcal{E}_{\text{LM}}(x) & \text{if } x \in \mathbb{L}_{\text{LM}} \\ \mathbf{W}\mathcal{E}_{\text{W2V}}(x) & \text{otherwise} \end{cases} \quad (1)$$

3.3 Updating the tokenizer

In a final step, we update the tokenizer to account for the added words. Let $\mathcal{T}_{\text{LM}}$ be the standard BERT tokenizer, and let $\hat{\mathcal{T}}_{\text{LM}}$ be the tokenizer that treats all words in $\mathbb{L}_{\text{LM}} \cup \mathbb{L}_{\text{W2V}}$ as one-wordpiece tokens, while tokenizing any other words as usual.

In practice, a given word may or may not benefit from being tokenized by $\hat{\mathcal{T}}_{\text{LM}}$ instead of $\mathcal{T}_{\text{LM}}$. To give a concrete example, 82% of the words in the BC5CDR NER dataset that end in the suffix *-ia* are part of a disease entity (e.g., *dementia*). $\mathcal{T}_{\text{LM}}$ tokenizes this word as *dem ##ent ##ia*, thereby exposing this strong orthographic cue to the model. As a result, $\mathcal{T}_{\text{LM}}$ improves recall on *-ia* diseases. But there are many cases where wordpiece tokenization is meaningless or misleading. For instance *euthymia* (not a disease) is tokenized by $\mathcal{T}_{\text{LM}}$ as *e ##uth ##ym ##ia*, making it likely to be classified as a disease. By contrast, $\hat{\mathcal{T}}_{\text{LM}}$ gives *euthymia* a one-wordpiece representation that depends only on distributional semantics. We find that using $\hat{\mathcal{T}}_{\text{LM}}$ improves precision on *-ia* diseases.

To combine these complementary strengths, we use a 50/50 mixture of $\mathcal{T}_{\text{LM}}$-tokenization and $\hat{\mathcal{T}}_{\text{LM}}$-tokenization when finetuning the PTLM on a task. At test time, we use both tokenizers and mean-pool the outputs. Let $o(S; \mathcal{T})$ be some output of interest (e.g., a logit), given sentence S tokenized by $\mathcal{T}$. We predict:

$$\hat{o}(S) = \frac{o(S; \mathcal{T}_{\text{LM}}) + o(S; \hat{\mathcal{T}}_{\text{LM}})}{2}$$

4 Experiment 1: Biomedical NER

In this section, we use the proposed method to create GreenBioBERT, an inexpensive and environmentally friendly alternative to BioBERT. Recall that BioBERTv1.0 (*biobert_v1.0_pubmed_pmc*) was initialized from general-domain BERT (*bert-base-cased*) and then pretrained on PubMed+PMC.

4.1 Domain adaptation

We train Word2Vec with vector size $d_{\text{W2V}} = d_{\text{LM}} = 768$ on PubMed+PMC (see Appendix for details). Then, we update the wordpiece embedding layer and tokenizer of general-domain BERT (*bert-base-cased*) as described in Section 3.

[2] Since we are not primarily interested in word alignment accuracy, we use the entire intersection as a training set in all other experiments.

Biomedical NER task	(NER task ID)	BERT (ref) (Lee et al., 2020)	BioBERTv1.0 (ref) (Lee et al., 2020)	BioBERTv1.1 (ref) (Lee et al., 2020)	GreenBioBERT (with standard error of the mean)
BC5CDR-disease (Li et al., 2016)	(1)	81.97 / 82.48 / 82.41	85.86 / 87.27 / 86.56	**86.47 / 87.84 / 87.15**	84.88 (.07) / 85.29 (.12) / 85.08 (.08)
NCBI-disease (Doğan et al., 2014)	(2)	84.12 / 87.19 / 85.63	**89.04** / 89.69 / 89.36	88.22 / **91.25 / 89.71**	85.49 (.23) / 86.41 (.15) / 85.94 (.16)
BC5CDR-chem (Li et al., 2016)	(3)	90.94 / 91.38 / 91.16	93.27 / **93.61** / 93.44	93.68 / 93.26 / **93.47**	**93.82** (.11) / 92.35 (.17) / 93.08 (.07)
BC4CHEMD (Krallinger et al., 2015)	(4)	91.19 / 88.92 / 90.04	92.23 / 90.61 / 91.41	**92.80 / 91.92 / 92.36**	**92.80** (.04) / 89.78 (.07) / 91.26 (.04)
BC2GM (Smith et al., 2008)	(5)	81.17 / 82.42 / 81.79	**85.16** / 83.65 / 84.40	84.32 / **85.12 / 84.72**	83.34 (.15) / 83.58 (.09) / 83.45 (.10)
JNLPBA (Kim et al., 2004)	(6)	69.57 / 81.20 / 74.94	**72.68** / 83.21 / **77.59**	72.24 / **83.56** / 77.49	71.93 (.12) / 82.58 (.12) / 76.89 (.10)
LINNAEUS (Gerner et al., 2010)	(7)	91.17 / 84.30 / 87.60	**93.84 / 86.11 / 89.81**	90.77 / 85.83 / 88.24	92.50 (.17) / 84.54 (.26) / 88.34 (.18)
Species-800 (Pafilis et al., 2013)	(8)	69.35 / 74.05 / 71.63	72.84 / **77.97 / 75.31**	72.80 / 75.36 / 74.06	**73.19** (.26) / 75.47 (.33) / 74.31 (.24)

Table 4: Biomedical NER test set precision / recall / F1 (%). "(ref)": Reference scores from Lee et al. (2020). **Boldface**: Best model in row. Underlined: Best model without target-domain LM pretraining.

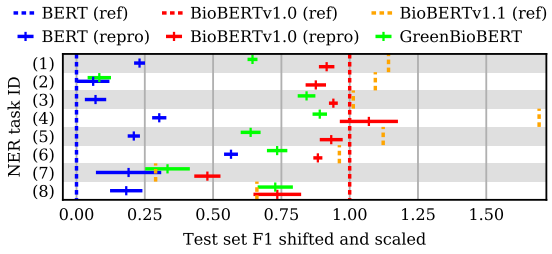

Figure 1: NER test set F1, transformed as $(x - \text{BERT}_{(\text{ref})})/(\text{BioBERTv1.0}_{(\text{ref})} - \text{BERT}_{(\text{ref})})$. This plot shows what portion of the reported BioBERT – BERT F1 delta is covered. "(ref)": Reference scores from Lee et al. (2020). "(repro)": Results of our reproduction experiments. Error bars: Standard error of the mean.

NER task ID	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
non-aligned	-4.88	-3.50	-4.13	-3.34	-2.34	-0.56	-0.84	-4.63
random init	-4.33	-3.60	-3.19	-3.19	-1.92	-0.50	-0.84	-3.58

Table 5: Absolute drop in dev set F1 when using non-aligned word vectors or randomly initialized word vectors, instead of aligned word vectors.

4.2 Finetuning

We finetune GreenBioBERT on the eight publicly available NER tasks used in Lee et al. (2020). We also do reproduction experiments with general-domain BERT and BioBERTv1.0, using the same setup as our model. See Appendix for details on preprocessing and hyperparameters. Since some of the datasets are sensitive to the random seed, we report mean and standard error over eight runs.

4.3 Results and discussion

Table 4 shows entity-level precision, recall and F1, as measured by the CoNLL NER scorer. For ease of visualization, Figure 1 shows test set F1 shifted and scaled as

$$f(x) = \frac{x - \text{BERT}_{(\text{ref})}}{\text{BioBERTv1.0}_{(\text{ref})} - \text{BERT}_{(\text{ref})}}$$

where $\text{BERT}_{(\text{ref})}$ and $\text{BioBERTv1.0}_{(\text{ref})}$ are reported scores from Lee et al. (2020). In other words, the figure shows what portion of the reported BioBERT – BERT F1 delta is covered by our less expensive GreenBioBERT model. On average, we cover between 61% and 70% of the delta (61% for BioBERTv1.0, 70% for BioBERTv1.1, and 61% if we take our reproduction experiments as reference points).

4.3.1 Ablation study

To test whether the improvements over general-domain BERT are due to the aligned Word2Vec vectors, or just to the availability of additional word vectors in general, we perform an ablation study where we replace the aligned vectors with their non-aligned counterparts (by setting $\mathbf{W} = \mathbf{1}$ in Eq. 1) or with randomly initialized vectors. Table 5 shows that dev set F1 drops on all datasets under these circumstances, i.e., vector space alignment seems to be important.

5 Experiment 2: Covid-19 QA

In this section, we use the proposed method to quickly adapt an existing general-domain QA model to an emerging target domain: the Covid-19 pandemic. Our baseline model is SQuADBERT,[3] an existing BERT model that was finetuned on the general-domain SQuAD dataset (Rajpurkar et al., 2016). We evaluate on Deepset-AI Covid-QA (Möller et al., 2020), a SQuAD-style dataset with 2019 annotated span-selection questions about 147 papers from CORD-19 (Covid-19 Open Research Dataset).[4] We assume that there is no labeled target-domain data for finetuning on the task, and instead use the entire Covid-QA dataset as a test set. This is a realistic setup for an emerging domain without annotated training data.

[3] www.huggingface.co/bert-large-uncased-whole-word-masking-finetuned-squad

[4] https://pages.semanticscholar.org/coronavirus-research

	domain adaptation corpus	size	EM	F1	substr
SQuADBERT	——		33.04	58.24	65.87
GreenCovid-SQuADBERT	CORD-19 only	2GB	**34.62**	60.09	**68.20**
	CORD-19+PubMed+PMC	94GB	34.32	**60.23**	68.00

Table 6: Results (%) on Deepset-AI Covid-QA. EM (exact answer match) and F1 (token-level F1 score) are evaluated with the SQuAD scorer. "substr": Predictions that are a substring of the gold answer. Much higher than EM, because many gold answers are not minimal answer spans (see Appendix, "Notes on Covid-QA", for an example).

5.1 Domain adaptation

We train Word2Vec with vector size $d_{\text{W2V}} = d_{\text{LM}} = 1024$ on CORD-19 and/or PubMed+PMC. The process takes less than an hour on CORD-19 and about one day on the combined corpus, again without the need for a GPU. Then, we update SQuADBERT's wordpiece embedding layer and tokenizer, as described in Section 3. We refer to the resulting model as GreenCovidSQuADBERT.

5.2 Results and discussion

Table 6 shows that GreenCovidSQuADBERT outperforms general-domain SQuADBERT on all measures. Interestingly, the small CORD-19 corpus is enough to achieve this result (compare "CORD-19 only" and "CORD-19+PubMed+PMC"), presumably because it is specific to the target domain and contains the Covid-QA context papers.

6 Conclusion

As a reaction to the trend towards high-resource models, we have proposed an inexpensive, CPU-only method for domain-adapting Pretrained Language Models: We train Word2Vec vectors on target-domain data and align them with the wordpiece vector space of a general-domain PTLM.

On eight biomedical NER tasks, we cover over 60% of the BioBERT – BERT F1 delta, at 5% of BioBERT's domain adaptation CO_2 footprint and 2% of its cloud compute cost. We have also shown how to rapidly adapt an existing BERT QA model to an emerging domain – the Covid-19 pandemic – without the need for target-domain Language Model pretraining or finetuning.

We hope that our approach will benefit practitioners with limited time or resources, and that it will encourage environmentally friendlier NLP.

Acknowledgements

This research was funded by Siemens AG. We thank our anonymous reviewers for their helpful comments.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, USA.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *EMNLP-IJCNLP*, pages 3606–3611, Hong Kong, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, USA.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show your work: Improved reporting of experimental results. In *EMNLP-IJCNLP*, pages 2185–2194, Hong Kong, China.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. LINNAEUS: a species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):85.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *EMNLP-IJCNLP*, pages 4229–4239, Hong Kong, China.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019a. ClinicalBERT: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

Kexin Huang, Abhishek Singh, Sitong Chen, Edward T Moseley, Chih-ying Deng, Naomi George, and Charlotta Lindvall. 2019b. Clinical XLNet: Modeling sequential clinical notes and predicting prolonged mechanical ventilation. *arXiv preprint arXiv:1912.11975*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The CHEMDNER corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):1–17.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in Adam.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. 2020. Covid-qa: A question & answer dataset for covid-19.

Evangelos Pafilis, Sune P Frankild, Lucia Fanini, Sarah Faulwetter, Christina Pavloudi, Aikaterini Vasileiadou, Christos Arvanitidis, and Lars Juhl Jensen. 2013. The SPECIES and ORGANISMS resources for fast and accurate identification of taxonomic names in text. *PloS one*, 8(6).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, Austin, USA.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of BioCreative II gene mention recognition. *Genome biology*, 9(2):S2.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *ACL*, pages 3645–3650, Florence, Italy.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

Hai Wang, Dian Yu, Kai Sun, Janshu Chen, and Dong Yu. 2019. Improving pre-trained multilingual models with vocabulary expansion. In *CoNLL*, pages 316–327, Hong Kong, China.

Inexpensive Domain Adaptation of Pretrained Language Models (Appendix)

Word2Vec training

We downloaded the PubMed, PMC and CORD-19 corpora from:

- https://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/ [20 January 2020, 68GB raw text]
- https://ftp.ncbi.nlm.nih.gov/pubmed/baseline/ [20 January 2020, 24GB raw text]
- https://pages.semanticscholar.org/coronavirus-research [17 April 2020, 2GB raw text]

We extract all abstracts and text bodies and apply the BERT basic tokenizer (a rule-based word tokenizer that standard BERT uses before wordpiece tokenization). Then, we train CBOW Word2Vec[5] with negative sampling. We use default parameters except for the vector size (which we set to $d_{\text{W2V}} = d_{\text{LM}}$).

Experiment 1: Biomedical NER

Pretrained models

General-domain BERT and BioBERTv1.0 were downloaded from:

- www.storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip
- www.github.com/naver/biobert-pretrained

Data

We downloaded the NER datasets by following instructions on www.github.com/dmis-lab/biobert#Datasets. For detailed dataset statistics, see Lee et al. (2020).

Preprocessing

We use Lee et al. (2020)'s preprocessing strategy: We cut all sentences into chunks of 30 or fewer whitespace-tokenized words (without splitting inside labeled spans). Then, we tokenize every chunk S with $\mathcal{T} = \mathcal{T}_{\text{LM}}$ or $\mathcal{T} = \hat{\mathcal{T}}_{\text{LM}}$ and add special tokens:

$$X = \textit{[CLS]}\ \mathcal{T}(S)\ \textit{[SEP]}$$

Word-initial wordpieces in $\mathcal{T}(S)$ are labeled as *B(egin)*, *I(nside)* or *O(utside)*, while non-word-initial wordpieces are labeled as *X(ignore)*.

[5] www.github.com/tmikolov/word2vec

Modeling, training and inference

We follow Lee et al. (2020)'s implementation (www.github.com/dmis-lab/biobert): We add a randomly initialized softmax classifier on top of the last BERT layer to predict the labels. We finetune the entire model to minimize negative log likelihood, with the AdamW optimizer (Loshchilov and Hutter, 2018) and a linear learning rate scheduler (10% warmup). All finetuning runs were done on a GeForce Titan X GPU (12GB).

At inference time, we gather the output logits of word-initial wordpieces only. Since the number of word-initial wordpieces is the same for $\mathcal{T}_{\text{LM}}(S)$ and $\hat{\mathcal{T}}_{\text{LM}}(S)$, this makes mean-pooling the logits straightforward.

Hyperparameters

We tune the batch size and peak learning rate on the development set (metric: F1), using the same hyperparameter space as Lee et al. (2020):

Batch size: $[10, 16, 32, 64]$[6]

Learning rate: $[1 \cdot 10^{-5}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}]$

We train for 100 epochs, which is the upper end of the 50–100 range recommended by the original authors. After selecting the best configuration for every task and model (see Table 7), we train the final model on the concatenation of training and development set, as was done by Lee et al. (2020). See Figure 2 for expected maximum development set F1 as a function of the number of evaluated hyperparameter configurations (Dodge et al., 2019).

Experiment 2: Covid-19 QA

Pretrained model

We downloaded the SQuADBERT baseline from:

- www.huggingface.co/bert-large-uncased-whole-word-masking-finetuned-squad

Data

We downloaded the Deepset-AI Covid-QA dataset from:

- www.github.com/deepset-ai/COVID-QA/blob/master/data/question-answering/COVID-QA.json [24 June 2020]

[6] Since LINNAEUS and BC4CHEM have longer maximum tokenized chunk lengths than the other datasets, our hardware was insufficient to evaluate batch size 64 on them.

At the time of writing, the dataset contains 2019 questions and gold answer spans. Every question is associated with one of 147 research papers (contexts) from CORD-19.[7] Since we do not do target-domain finetuning, we treat the entire dataset as a test set.

Preprocessing

We tokenize every question-context pair (Q, C) with $\mathcal{T} = \mathcal{T}_{\text{LM}}$ or $\mathcal{T} = \hat{\mathcal{T}}_{\text{LM}}$, which yields $(\mathcal{T}(Q), \mathcal{T}(C))$. Since $\mathcal{T}(C)$ is usually too long to be digested in a single forward pass, we define a sliding window with width and stride $N = \text{floor}(\frac{509 - |\mathcal{T}(Q)|}{2})$. At step n, the "active" window is between $a_n^{(l)} = (n-1)N + 1$ and $a_n^{(r)} = \min(|C|, nN)$. The input is defined as:

$$X^{(n)} = \textit{[CLS]}\ \mathcal{T}(Q)\ \textit{[SEP]}\ \mathcal{T}(C)_{a_n^{(l)} - p_n^{(l)} : a_n^{(r)} + p_n^{(r)}}\ \textit{[SEP]}$$

$p_n^{(l)}$ and $p_n^{(r)}$ are chosen such that $|X^{(n)}| = 512$, and such that the active window is in the center of the input (if possible).

Modeling and inference

Feeding $X^{(n)}$ into the QA model yields start logits $\mathbf{h'}^{(\text{start},n)} \in \mathbb{R}^{|X^{(n)}|}$ and end logits $\mathbf{h'}^{(\text{end},n)} \in \mathbb{R}^{|X^{(n)}|}$. We extract and concatenate the slices that correspond to the active windows of all steps:

$$\mathbf{h}^{(*)} \in \mathbb{R}^{|\mathcal{T}(C)|}$$
$$\mathbf{h}^{(*)} = [\mathbf{h'}^{(*,1)}_{a_1^{(l)}:a_1^{(r)}}; \ldots; \mathbf{h'}^{(*,n)}_{a_n^{(l)}:a_n^{(r)}}; \ldots]$$

Next, we map the logits from the wordpiece level to the word level. This allows us to mean-pool the outputs of $\mathcal{T}_{\text{LM}}$ and $\hat{\mathcal{T}}_{\text{LM}}$ even when $|\mathcal{T}_{\text{LM}}(C)| \neq |\hat{\mathcal{T}}_{\text{LM}}(C)|$.

Let c_i be a word in C and let $\mathcal{T}(C)_{j:j+|\mathcal{T}(c_i)|}$ be the corresponding wordpieces. The start and end logits of c_i are:

$$o_i^{(*)} = \max_{j \leq j' \leq j+|\mathcal{T}(c_i)|} [h_{j'}^{(*)}]$$

Finally, we return the answer span $C_{k:k'}$ that maximizes $o_k^{(\text{start})} + o_{k'}^{(\text{end})}$, subject to the constraints that k' does not precede k and the answer contains no more than 500 characters.

[7] www.github.com/deepset-ai/COVID-QA/issues/103

Notes on Covid-QA

There are some important differences between Covid-QA and SQuAD, which make the task challenging:

- The Covid-QA contexts are full documents rather than single paragraphs. Thus, the correct answer may appear several times, often with slightly different wordings. But only a single occurrence is annotated as correct, e.g.:

  Question: What was the prevalence of Coronavirus OC43 in community samples in Ilorin, Nigeria?

  Correct: 13.3% (95% CI 6.9-23.6%) *# from main text*

  Predicted: 13.3%, 10/75 *# from abstract*

- SQuAD gold answers are defined as the "shortest span in the paragraph that answered the question" (Rajpurkar et al., 2016, p. 4), but many Covid-QA gold answers are longer and contain non-essential context, e.g.:

  Question: When was the Middle East Respiratory Syndrome Coronavirus isolated first?

  Correct: (MERS-CoV) was first isolated in 2012, in a 60-year-old man who died in Jeddah, KSA due to severe acute pneumonia and multiple organ failure

  Predicted: 2012

These differences are part of the reason why the exact match score is lower than the word-level F1 score and the substring score (see Table 6, bottom, main paper).

Biomedical NER task	(ID)	BERT (repro) hyperparams	BERT (repro) dev set F1	BioBERTv1.0 (repro) hyperparams	BioBERTv1.0 (repro) dev set F1	GreenBioBERT hyperparams	GreenBioBERT dev set F1
BC5CDR-disease	(1)	$32, 3 \cdot 10^{-5}$	82.12	$10, 1 \cdot 10^{-5}$	85.15	$32, 1 \cdot 10^{-5}$	83.90
NCBI-disease	(2)	$32, 3 \cdot 10^{-5}$	87.52	$32, 1 \cdot 10^{-5}$	87.99	$10, 3 \cdot 10^{-5}$	88.43
BC5CDR-chem	(3)	$64, 3 \cdot 10^{-5}$	91.00	$32, 1 \cdot 10^{-5}$	93.36	$10, 1 \cdot 10^{-5}$	92.59
BC4CHEMD	(4)	$16, 1 \cdot 10^{-5}$	88.02	$32, 1 \cdot 10^{-5}$	89.35	$16, 1 \cdot 10^{-5}$	88.53
BC2GM	(5)	$32, 1 \cdot 10^{-5}$	83.91	$64, 3 \cdot 10^{-5}$	85.54	$64, 3 \cdot 10^{-5}$	84.25
JNLPBA	(6)	$32, 5 \cdot 10^{-5}$	85.18	$32, 5 \cdot 10^{-5}$	85.30	$10, 3 \cdot 10^{-5}$	85.10
LINNAEUS	(7)	$16, 1 \cdot 10^{-5}$	96.67	$32, 1 \cdot 10^{-5}$	97.22	$10, 1 \cdot 10^{-5}$	96.49
Species-800	(8)	$32, 1 \cdot 10^{-5}$	72.70	$32, 1 \cdot 10^{-5}$	77.34	$16, 1 \cdot 10^{-5}$	75.93

Table 7: Best hyperparameters (batch size, peak learning rate) and best dev set F1 per NER task and model. BERT (repro) and BioBERTv1.0 (repro) refer to our reproduction experiments.

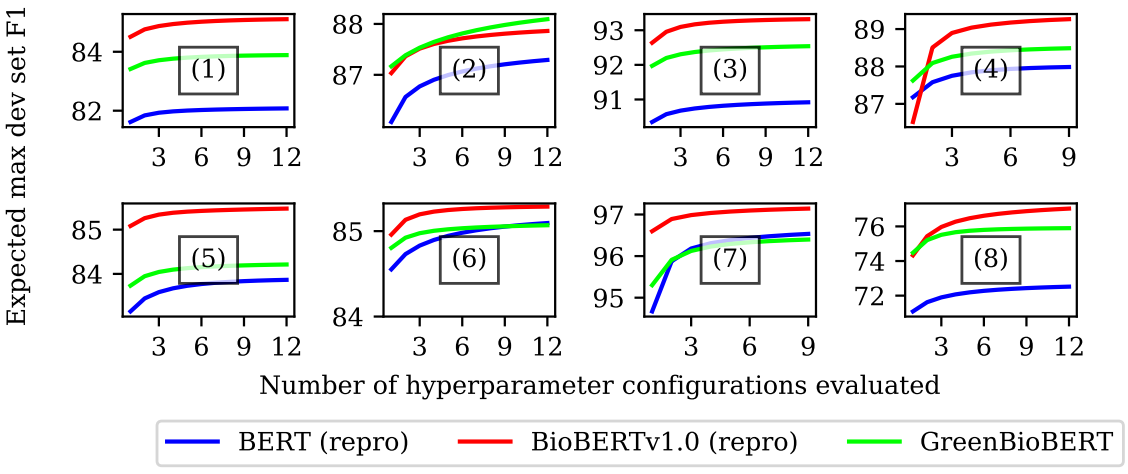

Figure 2: Expected maximum F1 on NER development sets as a function of the number of evaluated hyperparameter configurations. Numbers in brackets are NER task IDs (see Table 7).

Chapter 3

Corresponds to the following publication:

> Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (presented at Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures)*, pages 803–818, Online

Declaration of Co-Authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission. Ulli Waltinger contributed by reviewing an earlier draft of the paper.

[1] `https://www.aclweb.org/anthology/faq/copyright/` [accessed 05/03/2021]

E-BERT: Efficient-Yet-Effective Entity Embeddings for BERT

Nina Poerner[*†] **and Ulli Waltinger**[†] **and Hinrich Schütze**[*]
[*]Center for Information and Language Processing, LMU Munich, Germany
[†]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

Abstract

We present a novel way of injecting factual knowledge about entities into the pretrained BERT model (Devlin et al., 2019): We align Wikipedia2Vec entity vectors (Yamada et al., 2016) with BERT's native wordpiece vector space and use the aligned entity vectors as if they were wordpiece vectors. The resulting entity-enhanced version of BERT (called **E-BERT**) is similar in spirit to ERNIE (Zhang et al., 2019) and KnowBert (Peters et al., 2019), but it requires no expensive further pretraining of the BERT encoder. We evaluate E-BERT on unsupervised question answering (QA), supervised relation classification (RC) and entity linking (EL). On all three tasks, E-BERT outperforms BERT and other baselines. We also show quantitatively that the original BERT model is overly reliant on the surface form of entity names (e.g., guessing that someone with an Italian-sounding name speaks Italian), and that E-BERT mitigates this problem.

1 Introduction

BERT (Devlin et al., 2019) and its successors (e.g., Yang et al. (2019); Liu et al. (2019); Wang et al. (2019b)) continue to achieve state of the art performance on various NLP tasks. Recently, there has been interest in enhancing BERT with factual knowledge about entities (Zhang et al., 2019; Peters et al., 2019). To this end, we introduce **E-BERT**: We align Wikipedia2Vec entity vectors (Yamada et al., 2016) with BERT's wordpiece vector space (Section 3.1) and feed the aligned vectors into BERT as if they were wordpiece vectors (Section 3.2). Importantly, we do not make any changes to the BERT encoder itself, and we do no additional pretraining. This stands in contrast to previous entity-enhanced versions of BERT, such as ERNIE or KnowBert, which require additional encoder pretraining.

In Section 4, we evaluate our approach on LAMA (Petroni et al., 2019), a recent unsupervised QA benchmark for pretrained Language Models (LMs). We set a new state of the art on LAMA, with improvements over original BERT, ERNIE and KnowBert. We also find that the original BERT model is overly reliant on the surface form of entity names, e.g., it predicts that a person with an Italian-sounding name speaks Italian, regardless of whether this is factually correct. To quantify this effect, we create **LAMA-UHN** (UnHelpfulNames), a subset of LAMA where questions with overly helpful entity names were deleted (Section 4.4).

In Section 5, we show how to apply E-BERT to two entity-centric downstream tasks: relation classification (Section 5.1) and entity linking (Section 5.2). On the former task, we feed aligned entity vectors as inputs, on the latter, they serve as inputs *and* outputs. In both cases, E-BERT outperforms original BERT and other baselines.

Summary of contributions.

- Introduction of E-BERT: Feeding entity vectors into BERT without additional encoder pretraining. (Section 3)
- Evaluation on the LAMA unsupervised QA benchmark: E-BERT outperforms BERT, ERNIE and KnowBert. (Section 4)
- LAMA-UHN: A harder version of the LAMA benchmark with less informative entity names. (Section 4.4)
- Evaluation on supervised relation classification (Section 5.1) and entity linking (Section 5.2).
- Upon publication, we will release LAMA-UHN as well as E-BERT$_{\texttt{BASE}}$ and E-BERT$_{\texttt{LARGE}}$.[1]

[1] https://github.com/npoe/ebert

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 803–818
November 16 - 20, 2020.

2 Related work

2.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer (Vaswani et al., 2017) that was pretrained as a masked LM (MLM) on unlabeled text. At its base, BERT segments text into wordpieces from a vocabulary $\mathbb{L}_{\text{WP}}$. Wordpieces are embedded into real-valued vectors by a lookup function (denoted $\mathcal{E}_{\text{BERT}} : \mathbb{L}_{\text{WP}} \rightarrow \mathbb{R}^{d_{\text{BERT}}}$). The wordpiece vectors are combined with position and segment embeddings and then fed into a stack of Transformer layers (the encoder, denoted $\mathcal{F}_{\text{BERT}}$). During pretraining, some wordpieces are replaced by a special *[MASK]* token. The output of BERT is fed into a final feed-forward net (the MLM head, denoted $\mathcal{F}_{\text{MLM}}$), to predict the identity of the masked wordpieces. After pretraining, the MLM head is usually replaced by a task-specific layer, and the entire model is finetuned on supervised data.

2.2 Entity-enhanced BERT

This paper adds to recent work on entity-enhanced BERT models, most notably ERNIE (Zhang et al., 2019) and KnowBert (Peters et al., 2019). ERNIE and KnowBert are based on the design principle that *BERT be adapted to entity vectors*: They introduce new encoder layers to feed pretrained entity vectors into the Transformer, and they require additional pretraining to integrate the new parameters. In contrast, E-BERT's design principle is that *entity vectors be adapted to BERT*, which makes our approach more efficient (see Section 3.3).

Two other knowledge-enhanced MLMs are KEPLER (Wang et al., 2019c) and K-Adapter (Wang et al., 2020), which are based on Roberta (Liu et al., 2019) rather than BERT. Their factual knowledge does not stem from entity vectors – instead, they are trained in a multi-task setting on relation classification and knowledge base completion.

2.3 Wikipedia2Vec

Wikipedia2Vec (Yamada et al., 2016) embeds words and entities (Wikipedia URLs) into a common space. Given a vocabulary of words $\mathbb{L}_{\text{Word}}$ and a vocabulary of entities $\mathbb{L}_{\text{Ent}}$, it learns a lookup embedding function $\mathcal{E}_{\text{Wikipedia}} : \mathbb{L}_{\text{Word}} \cup \mathbb{L}_{\text{Ent}} \rightarrow \mathbb{R}^{d_{\text{Wikipedia}}}$. The Wikipedia2Vec loss has three components: (1) skipgram Word2Vec (Mikolov et al., 2013a) operating on $\mathbb{L}_{\text{Word}}$, (2) a graph loss operating on the Wikipedia hyperlink graph, whose vertices are $\mathbb{L}_{\text{Ent}}$ and (3) a version of Word2Vec where words are predicted from entities. Loss (3) ensures that entities and words are embedded into the same space.

2.4 Vector space alignment

Our vector space alignment strategy is inspired by cross-lingual word vector alignment (e.g., Mikolov et al. (2013b); Smith et al. (2017)). A related method was recently applied by Wang et al. (2019a) to map cross-lingual word vectors into the multilingual BERT wordpiece vector space.

2.5 Unsupervised QA

QA has typically been tackled as a supervised problem (e.g., Das et al. (2017); Sun et al. (2018)). Recently, there has been interest in using unsupervised LMs such as GPT-2 or BERT for this task (Radford et al., 2019; Petroni et al., 2019). Davison et al. (2019) mine unsupervised commonsense knowledge from BERT, and Jiang et al. (2019) show the importance of using good prompts for unsupervised QA. None of this prior work differentiates quantitatively between factual knowledge of LMs and their ability to reason about the surface form of entity names.

3 E-BERT

3.1 Aligning entity and wordpiece vectors

Conceptually, we want to transform the vectors of the entity vector space $\mathcal{E}_{\text{Wikipedia}}[\mathbb{L}_{\text{Ent}}]$ in such a way that they look to BERT like vectors from its native wordpiece vector space $\mathcal{E}_{\text{BERT}}[\mathbb{L}_{\text{WP}}]$. We model the transformation as an unconstrained linear mapping $\mathbf{W} \in \mathbb{R}^{d_{\text{BERT}} \times d_{\text{Wikipedia}}}$. Since $\mathbb{L}_{\text{WP}}$ does not contain any entities (i.e., $\mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Ent}} = \{\}$), we fit the mapping on $\mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Word}}$:

$$\sum_{x \in \mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Word}}} ||\mathbf{W}\mathcal{E}_{\text{Wikipedia}}(x) - \mathcal{E}_{\text{BERT}}(x)||_2^2$$

Since Wikipedia2Vec embeds $\mathbb{L}_{\text{Word}}$ and $\mathbb{L}_{\text{Ent}}$ into the same space (see Section 2.3), $\mathbf{W}$ can be applied to $\mathbb{L}_{\text{Ent}}$ as well. We define the E-BERT embedding function as:

$$\mathcal{E}_{\text{E-BERT}} : \mathbb{L}_{\text{Ent}} \rightarrow \mathbb{R}^{d_{\text{BERT}}}$$
$$\mathcal{E}_{\text{E-BERT}}(a) = \mathbf{W}\mathcal{E}_{\text{Wikipedia}}(a)$$

Table 1 shows that despite its simplicity, the linear mapping achieves high alignment accuracies on seen and unseen vector pairs.

	Acc@1	Acc@5	Acc@10
train (19.6K words)	90.9	95.7	96.6
development (2.2K words)	83.0	90.9	92.6

Table 1: $\mathbb{L}_{\text{Word}} \rightarrow \mathbb{L}_{\text{WP}}$ alignment accuracy (%), i.e., how often the correct wordpiece vector is among the top-K Nearest Neighbors (by cosine) of an aligned Wikipedia2Vec word vector. In this table, we hold out 10 % of $\mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Word}}$ as a development set. In all other experiments, we fit $\mathbf{W}$ on the entire intersection.

3.2 Using aligned entity vectors

We explore two strategies for feeding the aligned entity vectors into the BERT encoder:

E-BERT-concat. E-BERT-concat combines entity IDs and wordpieces by string concatenation, with the slash symbol as separator (Schick and Schütze, 2019). For example, the wordpiece-tokenized input

The native language of Jean Mara ##is is [MASK] .[2]

becomes

The native language of **Jean_Marais** */ Jean Mara ##is is [MASK] .*

The entity ID (**bold**) is embedded by $\mathcal{E}_{\text{E-BERT}}$ and all wordpieces (*italics*) are embedded by $\mathcal{E}_{\text{BERT}}$ (see Figure 1). After the embedding operation, the sequence of vectors is combined with position and segment embeddings and fed into $\mathcal{F}_{\text{BERT}}$, just like any normal sequence of wordpiece vectors.

E-BERT-concat is comparable to ERNIE or KnowBert, which also represent entities as a combination of surface form (wordpieces) and entity vectors. But in contrast to ERNIE and KnowBERT, **we do not change or further pretrain the BERT encoder itself**.

E-BERT-replace. For ablation purposes, we define another variant of E-BERT that substitutes the entity surface form with the entity vector. With E-BERT-replace, our example becomes:

The native language of **Jean_Marais** *is [MASK] .*

A note on entity links. So far, we assume that we know which Wikipedia entity ID a given string refers to, i.e., that we have access to gold entity links. Depending on the nature of the task, these gold entity links may be given as part of the dataset (RC task), or they may be heuristically annotated

[2]For readability, we omit the special tokens *[CLS]* and *[SEP]* from all examples.

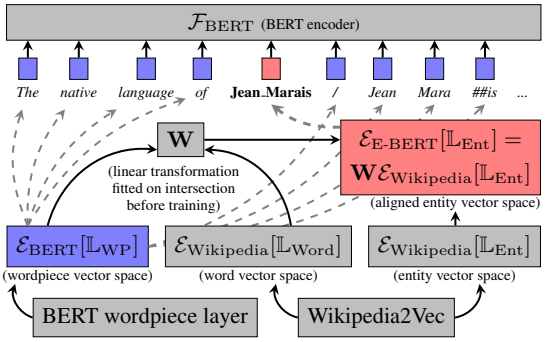

Figure 1: Schematic depiction of E-BERT-concat.

(see Appendix on how to reverse-map LAMA entity names). In other scenarios, we need an entity linker. In this respect, E-BERT is comparable to ERNIE but not to KnowBert, which has a built-in latent entity linker. Alternatively, we can train E-BERT as an entity linker first (see Section 5.2) and then use the resulting model to annotate training data for a different task.

3.3 Implementation

We train cased Wikipedia2Vec on a recent Wikipedia dump (2019-09-02), setting $d_{\text{Wikipedia}} = d_{\text{BERT}}$. We ignore Wikipedia pages with fewer than 5 links (Wikipedia2Vec's default), with the exception of entities needed for the downstream entity linking experiments (see Section 5.2). This results in an entity vocabulary of size $|\mathbb{L}_{\text{Ent}}| = 2.7\text{M}$.[3]

Computational cost. Training Wikipedia2Vec took us ∼6 hours on 32 CPUs, and the cost of fitting the linear transformation $\mathbf{W}$ is negligible. We did not require a GPU. For comparison, KnowBert W+W was pretrained for 1.25M steps on up to four Titan RTX GPUs, and ERNIE took one epoch on the English Wikipedia. (ERNIE's pretraining hardware was not disclosed, but it seems likely that a GPU was involved.)

4 Unsupervised QA

4.1 Data

The LAMA (LAnguage Model Analysis) benchmark (Petroni et al., 2019) probes for "factual and commonsense knowledge" of pretrained LMs. In

[3]Due to the link threshold and some Wikidata-Wikipedia mismatches, we lack entity vectors for 6% of LAMA questions and 10% of FewRel sentences (RC experiment, see Section 5.1). In these cases, we fall back onto using wordpieces only, i.e., onto standard BERT behavior.

this paper, we use LAMA-Google-RE and LAMA-T-REx (Elsahar et al., 2018), which are aimed at factual knowledge. Contrary to most previous work on QA, LAMA tests LMs without supervised fine-tuning. Petroni et al. (2019) claim that BERT's performance on LAMA is comparable with a knowledge base (KB) automatically extracted from text, and speculate that BERT and similar models "might become a viable alternative" to such KBs.

The LAMA task follows this schema: Given a KB triple (**sub**, **rel**, **obj**), the object is elicited with a relation-specific cloze-style question, e.g., (**Jean_Marais**, **native-language**, **French**) becomes: "The native language of Jean Marais is [MASK]."[4] The model predicts a probability distribution over a limited vocabulary $\mathbb{L}_{\text{LAMA}} \subset \mathbb{L}_{\text{WP}}$ to replace [MASK], which is evaluated against the surface form of the object (here: *French*).

4.2 Baselines

Our primary baselines are cased $\text{BERT}_{\texttt{BASE}}$ and $\text{BERT}_{\texttt{LARGE}}$[5] as evaluated in Petroni et al. (2019). We also test ERNIE (Zhang et al., 2019)[6] and KnowBert W+W (Peters et al., 2019),[7] two entity-enhanced $\text{BERT}_{\texttt{BASE}}$-type models.[8] E-BERT, ERNIE and KnowBert have entity vocabularies of size 2.7M, 5M and 470K, respectively. As this might put KnowBert at a disadvantage, Table 4 also reports performance on the subset of questions whose gold subject is known to KnowBert.

4.3 Evaluation measure

We use the same evaluation measure as Petroni et al. (2019): For a given k, we count a question as 1 if the correct answer is among the top-k predictions and as 0 otherwise. Petroni et al. (2019) call this measure Precision@k (P@k). Since this is not in line with the typical use of the term "precision" in information retrieval (Manning et al., 2008, p. 161), we call the evaluation measure Hits@k. Like Petroni et al. (2019), we first average within relations and then across relations.

	original BERT	E-BERT-replace	E-BERT-concat	ERNIE	Know-Bert
Jean Marais	French	French	French	french	french
Daniel Ceccaldi	Italian	French	French	french	italian
Orane Demazis	Albanian	French	French	french	french
Sylvia Lopez	Spanish	French	Spanish	spanish	spanish
Annick Alane	English	French	French	english	english

Table 2: Native language (LAMA-T-REx:P103) of French-speaking actors according to different models. Model size is `BASE`.

4.4 LAMA-UHN

Imagine a person who claims to know a lot of facts. During a quiz, you ask them about the native language of actor Jean Marais. They correctly answer "French." For a moment you are impressed, until you realize that Jean is a typical French name. So you ask the same question about Daniel Ceccaldi (a French actor with an Italian-sounding name). This time, the person says "Italian."

If this quiz were a QA benchmark, the person would have achieved a respectable Hits@1 score of 50%. Yet, you doubt that they really *knew* the first answer.

Qualitative inspection of BERT's answers to LAMA suggests that the model often behaves less like a KB and more like the person just described. In Table 2 for instance, BERT predicts native languages that are plausible for people's names, even when there is no factual basis for these predictions. This kind of name-based reasoning is a useful strategy for getting a high score on LAMA, as the correct answer and the best name-based guess tend to coincide (e.g., people with Italian-sounding names frequently speak Italian). Hence, LAMA in its current form cannot differentiate whether a model is good at reasoning about (the surface form of) entity names, good at memorizing facts, or both. To quantify the effect, we create LAMA-UHN (UnHelpful Names), a subset of LAMA where overly helpful entity names are heuristically deleted:

Heuristic 1 (string match filter). We first delete all KB triples (questions) where the correct answer (e.g., *Apple*) is a case-insensitive substring of the subject entity name (e.g., *Apple Watch*). This affects 12% of all triples, and up to 81% for individual relations (see Table 3, top).

[4] LAMA provides oracle entity IDs, however, they are not used by the BERT baseline. For a fair evaluation, we ignore them too and instead use the Wikidata query API (`https://query.wikidata.org`) to infer entity IDs from surface forms. See Appendix for details.

[5] `https://github.com/huggingface/transformers`

[6] `https://github.com/thunlp/ERNIE`

[7] `https://github.com/allenai/kb`

[8] ERNIE and KnowBert are uncased models. We therefore lowercase all questions for them and restrict predictions to the intersection of their wordpiece vocabulary with lowercased $\mathbb{L}_{\text{LAMA}}$. As a result, ERNIE and KnowBert select answers from ∼18K candidates (instead of ∼21K), which should work in their favor. We verify that all lowercased answers appear in this vocabulary, i.e., ERNIE and KnowBert are in principle able to answer all questions correctly.

Heuristic		Relation	% deleted	Example of a deleted question	
1	string match filter	T-REx:P176 (manufacturer)	81%	Fiat Multipla is produced by [MASK:Fiat].	
		T-REx:P138 (named after)	75%	Christmas Island is named after [MASK:Christmas].	
		T-REx:P1001 (applies to jurisdiction)	73%	Australian Senate is a legal term in [MASK:Australia].	
		T-REx:P279 (subclass of)	51%	lenticular galaxy is a subclass of [MASK:galaxy].	
		T-REx:P31 (instance of)	39%	[Tantalon Castle] is a [MASK:castle].	
2	person name filter	T-REx:P1412 (language used)	63%	Fulvio Tomizza used to communicate in [MASK:Italian].	(1,1)
		T-REx:P103 (native language)	58%	The native language of Tommy Nilsson is [MASK:Swedish].	(-,1)
		T-REx:P27 (nationality)	56%	Harumi Inoue is a [MASK:Japan] citizen.	(1,-)
		T-REx:P20 (place of death)	31%	Avraham Harman died in [MASK:Jerusalem].	(1,-)
		T-REx:P19 (place of birth)	23%	[Christel Bodenstein] was born in [MASK:Munich].	(3,3)

Table 3: Statistics and examples of LAMA questions with helpful entity names, which were deleted from LAMA-UHN. We show the top-5 most strongly affected relations per heuristic. Numbers in brackets indicate which part(s) of the person name triggered the person name filter, e.g., (-,1) means that the correct answer was ranked first for the person's last name, but was not in the top-3 for their first name.

	Model size	BASE					LARGE			
	Model / Dataset	original BERT	E-BERT-replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	K-Adapter
All subjects	0 (original LAMA)	29.2	29.1	**36.2**	30.4	31.7	30.6	28.5	34.2	27.6
	1	22.3	29.2	**32.6**	25.5	25.6	24.6	28.6	30.8	-
	2 (LAMA-UHN)	20.2	28.2	**31.1**	24.7	24.6	23.0	27.8	29.5	21.7
	LAMA-UHN complement	52.7	25.9	**56.8**	36.2	47.0	52.7	32.1	34.5	-
KnowBert subjects only	0 (original LAMA)	32.0	28.5	**35.8**	30.4	32.0	33.1	28.2	34.9	-
	1	24.8	28.6	**32.0**	25.7	25.9	27.0	28.3	31.5	-
	2 (LAMA-UHN)	22.8	27.7	**30.6**	24.9	25.1	25.5	27.4	**30.6**	-

Table 4: Mean Hits@1 on LAMA-Google-RE and LAMA-T-REx combined. 0: original LAMA dataset (Petroni et al., 2019), 1: after string match filter, 2: after string match filter and person name filter (LAMA-UHN). "LAMA-UHN complement": Evaluating on all questions that were deleted from LAMA-UHN. "KnowBert subjects only": Evaluating on questions whose gold subject is in the KnowBert entity vocabulary. Results for K-Adapter are calculated from Wang et al. (2020, Table 5). See Appendix for individual relations.

Heuristic 2 (person name filter). Entity names can be revealing in ways that are more subtle than string matches. As illustrated by our *Jean Marais* example, a person's name can be a useful prior for guessing their native language and by extension, their nationality, place of birth, etc. We therefore use cloze-style questions to elicit name associations inherent in BERT, and delete triples that correlate with them.

The heuristic is best explained via an example. Consider again (**Jean_Marais**, **native-language**, **French**). We whitespace-tokenize the subject's surface form *Jean Marais* into *Jean* and *Marais*. If BERT considers either name to be a common French name, then a correct answer is insufficient evidence for factual knowledge about the entity **Jean_Marais**. On the other hand, if neither *Jean* nor *Marais* are considered French, but a correct answer is given regardless, we consider it sufficient evidence of factual knowledge.

We query BERT with "[X] is a common name in the following language: [MASK]." for [X] = *Jean* and [X] = *Marais*. (Depending on the relation, we replace "language" with "city" or "country".) If *French* is among the top-3 answers for either question, we delete the original triple. We apply this heuristic to T-REx:P19 (place of birth), T-REx:P20 (place of death), T-REx:P27 (nationality), T-REx:P103 (native language), T-REx:P1412 (language used), Google-RE:place-of-death and Google-RE:place-of-birth. See Table 3 (bottom) for examples and statistics.

4.5 Results and discussion

Table 4 shows mean Hits@1 on the original LAMA dataset (0), after applying the string match filter (1), and after applying both filters (2, LAMA-UHN). We also show mean Hits@1 on the LAMA-UHN complement, i.e., on the set of all questions with helpful entity names.

E-BERT-concat$_{\text{BASE}}$ sets a new state of the art on LAMA, with major gains over original BERT. To understand why, compare the performances of original BERT$_{\text{BASE}}$ and E-BERT-replace$_{\text{BASE}}$ on LAMA-UHN and the LAMA-UHN complement: On LAMA-UHN, BERT$_{\text{BASE}}$ drops by 9% (relative to original LAMA), while E-BERT-replace$_{\text{BASE}}$ drops by less than 1%. On the comple-

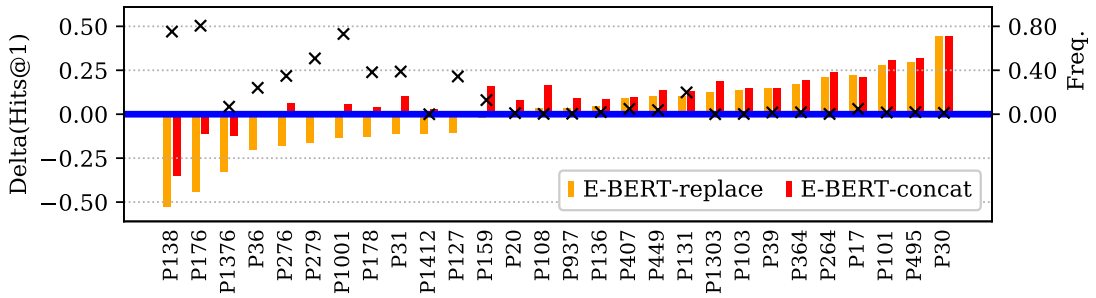

Figure 2: Left y-axis (bars): delta in mean Hits@1 relative to BERT on individual LAMA relations. Right y-axis (crosses): frequency of questions where the answer is a substring of the subject entity name (i.e., questions that would be deleted by the string match filter). Model size: BASE. Due to space constraints, we only show relations with max absolute delta ≥ 0.075.

ment, $\text{BERT}_{\text{BASE}}$ gains over 20%, while $\text{E-BERT-replace}_{\text{BASE}}$ drops slightly. This suggests that BERT's performance on original LAMA is partly due to the exploitation of helpful entity names, while that of E-BERT-replace is due to factual knowledge. Since $\text{E-BERT-concat}_{\text{BASE}}$ has access to entity names *and* entity vectors, it can leverage and combine these complementary sources of information.

For a more in-depth analysis, Figure 2 shows Delta(Hits@1) w.r.t. BERT (bars, left axis) on individual relations, along with the frequency of questions whose correct answer is a substring of the subject name (crosses, right axis). The losses of E-BERT-replace are almost exclusively on relations with a high frequency of "easy" substring answers, while its gains are on relations where such answers are rare. E-BERT-concat mitigates most of the losses of E-BERT-replace while keeping most of its gains. Figure 3 shows that gains of E-BERT-concat over BERT, KnowBert and ERNIE in terms of mean Hits@k are especially big for $k > 1$. This means that while E-BERT-concat is moderately better than the baselines at giving the correct answer, it is a lot better at "almost giving the correct answer". Petroni et al. (2019) speculate that even when factual knowledge is not salient enough for a top-1 answer, it may still be useful when finetuning on a downstream task.

5 Downstream tasks

We now demonstrate how to use E-BERT on two downstream tasks: relation classification (RC) and entity linking (EL). In both experiments, we keep the embedding layer ($\mathcal{E}_{\text{BERT}}$ and/or $\mathcal{E}_{\text{E-BERT}}$) fixed but finetune all other encoder parameters. We use the $\text{BERT}_{\text{BASE}}$ architecture throughout.

5.1 Relation classification

In relation classification (RC), a model learns to predict the directed relation of entities a_{sub} and a_{obj} from text. For instance, given the sentence

Taylor was later part of the ensemble cast in MGM 's classic World War II drama " Battleground " (1949) .

with surface forms *Battleground* and *World War II* referring to a_{sub} = **Battleground_(film)** and a_{obj} = **Word_War_II**, the model should predict the relation **primary-topic-of-work**. We have three ways of embedding this example:

original BERT (wordpieces): *[...] classic World War II drama " Battle ##ground " (1949) .*

E-BERT-concat: *[...] classic* **World_War_II** */ World War II drama "* **Battleground_(film)** */ Battle ##ground " (1949) .*

E-BERT-replace: *[...] classic* **World_War_II** *drama "* **Battleground_(film)** *" (1949) .*

As before, entity IDs (**bold**) are embedded by $\mathcal{E}_{\text{E-BERT}}$ and wordpieces (*italics*) by $\mathcal{E}_{\text{BERT}}$.

Baselines. To assess the impact of vector space alignment, we train two additional models (Wikipedia2Vec-BERT-concat and Wikipedia2Vec-BERT-replace) that feed non-aligned Wikipedia2Vec vectors directly into BERT (i.e., they use $\mathcal{E}_{\text{Wikipedia}}$ instead of $\mathcal{E}_{\text{E-BERT}}$ to embed entity IDs).

Data. We evaluate on a preprocessed dataset from Zhang et al. (2019), which is a subset of the FewRel corpus (Sun et al., 2018) (see Appendix for details). We use the FewRel oracle entity IDs, which are also used by ERNIE. Our entity coverage is lower than ERNIE's (90% vs. 96%), which should put us at a disadvantage. See Appendix for details on data and preprocessing.

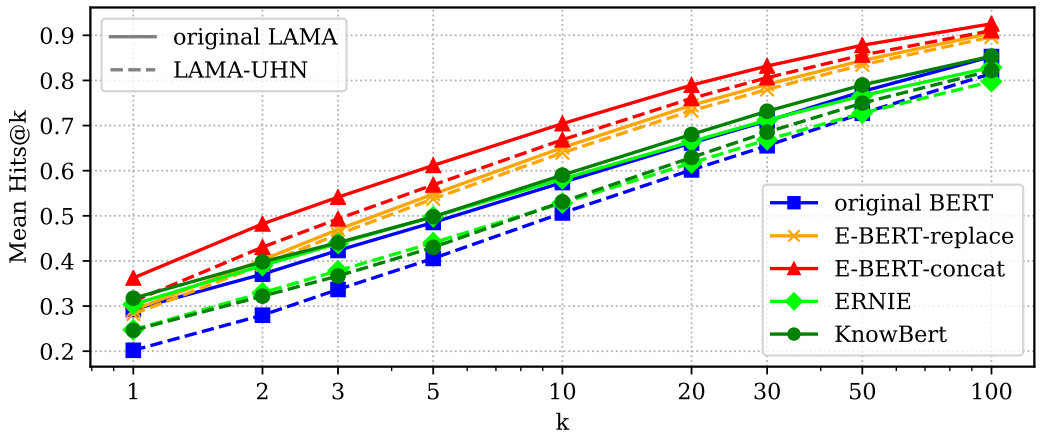

Figure 3: Mean Hits@k for different k. Model size: BASE. The x-axis is on a logarithmic scale.

	dev set			test set		
	P	R	F1	P	R	F1
original BERT	85.88	85.81	85.75	85.57	85.51	85.45
E-BERT-concat	**88.35**	**88.29**	**88.19**	**88.51**	**88.46**	**88.38**
E-BERT-replace	87.24	87.15	87.09	87.34	87.33	87.22
Wikipedia2Vec-BERT-concat	85.96	85.71	85.69	85.94	85.93	85.84
Wikipedia2Vec-BERT-replace	77.25	77.11	77.07	77.63	77.52	77.45
ERNIE (Zhang et al., 2019)	-	-	-	**88.49**	**88.44**	**88.32**

Table 5: RC macro precision, recall and F1 (%).

Modeling and hyperparameters. We adopt the setup and hyperparameters of Zhang et al. (2019): We use the *#* and *$* tokens to mark subject and object spans in the input, and we feed the last contextualized vector of the *[CLS]* token into a randomly initialized softmax classifier. Like Zhang et al. (2019), we use the Adam optimizer (Kingma and Ba, 2014) with a linear learning rate scheduler (10% warmup) and a batch size of 32. We tune the number of training epochs and the peak learning rate on the same parameter ranges as Zhang et al. (2019). See Appendix for details.

Results and discussion. E-BERT-concat performs better than original BERT and slightly better than ERNIE (Table 5). Recall that ERNIE required additional encoder pretraining to achieve this result. Interestingly, E-BERT-replace (which is entity-only) beats original BERT (which is surface-form-only), i.e., aligned entity vectors seem to be more useful than entity names for this task. The drop in F1 from E-BERT to Wikipedia2Vec-BERT shows the importance of vector space alignment.

5.2 Entity linking

Entity linking (EL) is the task of detecting entity spans in a text and linking them to the underlying entity ID. While there are recent advances in fully end-to-end EL (Broscheit, 2019), the task is typically broken down into three steps: (1) detecting spans that are potential entity spans, (2) generating sets of candidate entities for these spans, (3) selecting the correct candidate for each span.

For steps (1) and (2), we use KnowBert's candidate generator (Peters et al., 2019), which is based on a precomputed span-entity co-occurrence table (Hoffart et al., 2011). Given an input sentence, the generator finds all spans that occur in the table, and annotates each with a set of candidates $A = \{a_1 \dots a_N\}$ and prior probabilities $\{p(a_1) \dots p(a_N)\}$. Note that the candidates and priors are span- but not context-specific, and that the generator may over-generate. For step (3), our model must therefore learn to (a) reject over-generated spans and (b) disambiguate candidates based on context.

Modeling. Recall that BERT was pretrained as a masked LM (MLM). Given a wordpiece-tokenized input X with $x_i = \textit{[MASK]}$, it predicts a probability distribution over $\mathbb{L}_{\text{WP}}$ to replace x_i:

$$p(w|X) \propto \exp(\mathbf{e}_w \cdot \mathcal{F}_{\text{MLM}}(\mathbf{h}_i) + b_w) \qquad (1)$$

where $\mathbf{h}_i$ is the contextualized embedding of *[MASK]*, b_w is a learned bias and $\mathbf{e}_w = \mathcal{E}_{\text{BERT}}(w)$. (See also Section 2.1 for notation.) Since $\mathcal{E}_{\text{E-BERT}}[\mathbb{L}_{\text{Ent}}]$ is aligned with $\mathcal{E}_{\text{BERT}}[\mathbb{L}_{\text{WP}}]$, the pretrained MLM should have a good initialization for predicting entities from context as well.

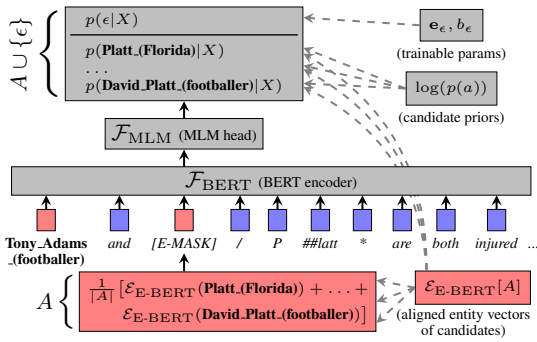

Figure 4: Schematic depiction of E-BERT-MLM in inference mode, predicting an entity vector for the name "Platt" in context. Blue: $\mathcal{E}_{\text{BERT}}$ wordpiece vectors. Red: $\mathcal{E}_{\text{E-BERT}}$ entity vectors. The candidates A and their priors $p(a)$ are given by the candidate generator. Assume that the entity **Tony_Adams_(footballer)** was decoded in a previous iteration (see "Iterative refinement").

Based on this intuition, our **E-BERT-MLM** model repurposes the MLM for the entity selection step. Given a wordpiece-tokenized span $s_1 \ldots s_{T_s}$ with left context $l_1 \ldots l_{T_l}$, right context $r_1 \ldots r_{T_r}$, candidates A and priors $p(a)$, we define:

$$X = l_1 \ldots l_{T_l}\ \textit{[E-MASK]}\ /\ s_1 \ldots s_{T_s} * r_1 \ldots r_{T_r}$$

All tokens in X except *[E-MASK]* are embedded by $\mathcal{E}_{\text{BERT}}$. *[E-MASK]* is embedded as $\frac{1}{|A|}\sum_{a \in A} \mathcal{E}_{\text{E-BERT}}(a)$, to inform the encoder about its options for the current span. (See Table 6 for an ablation with the standard *[MASK]* token.)

The output probability distribution for *[E-MASK]* is not defined over $\mathbb{L}_{\text{WP}}$ but over $A \cup \{\epsilon\}$, where ϵ stands for rejected spans (see below):

$$p(a|X) \propto \exp(\mathbf{e}_a \cdot \mathcal{F}_{\text{MLM}}(\mathbf{h}_{T_l+1}) + b_a) \quad (2)$$

where $\mathbf{e}_a = \mathcal{E}_{\text{E-BERT}}(a)$ and $b_a = \log(p(a))$.[9] The null-entity ϵ has parameters $\mathbf{e}_\epsilon, b_\epsilon$ that are trained from scratch.

Finetuning. We finetune E-BERT-MLM on the training set to minimize $\sum_{(X,\hat{a})} -\log(p(\hat{a}|X))$, where $(X, \hat{a})$ are pairs of potential spans and their gold entities. If X has no gold entity (if it was over-generated), then $\hat{a} = \epsilon$.[10]

[9] To understand why we set $b_a = \log(p(a))$, assume that the priors are implicitly generated as $p(a) = \exp(b_a)/Z$, with $Z = \sum_{a'} \exp(b_{a'})$. It follows that $b_a = \log(p(a)) + \log(Z)$. Since $\log(Z)$ is the same for all a', and the softmax function is invariant to constant offsets, we can drop $\log(Z)$ from Eq. 2.

[10] If $\hat{a} \neq \epsilon \wedge \hat{a} \notin A$, we remove the span from the training set. We do **not** do this at test time, i.e., we evaluate on all gold standard entities.

	AIDA-A (dev)		AIDA-B (test)	
	Micro	Macro	Micro	Macro
E-BERT-MLM	**90.8**	**89.1**	**85.0**	**84.2**
w/o iterative refinement	90.6	89.0	-	-
w/ standard *[MASK]* token	90.3	88.8	-	-
Wikipedia2Vec-BERT-MLM	88.7	86.4	80.6	81.0
Wikipedia2Vec-BERT-random	88.2	86.1	80.5	81.2
Kolitsas et al. (2018)	89.4	86.6	82.4	82.6
Broscheit (2019)	86.0	-	79.3	-
KnowBert (Peters et al., 2019)	82.1	-	73.7	-
Chen et al. (2019)†	**92.6**	**93.6**	**87.5**	**87.7**

Table 6: F1 (%) on AIDA after finetuning. †Might not be comparable: Chen et al. (2019) evaluate on in-vocabulary entities only, without ensuring (or reporting) the vocabulary's coverage of the AIDA data.

Iterative refinement. We found it useful to iteratively refine predictions during inference, similar to techniques from non-autoregressive Machine Translation (Ghazvininejad et al., 2019). We start with a wordpiece-tokenized input, e.g.:

> *Adams and P ##latt are both injured and will miss England 's opening World Cup qualifier ...*

We make predictions for all potential spans that the candidate generator finds in the input. We gather all spans with $\text{argmax}_a[p(a|X)] \neq \epsilon$, sort them by $1 - p(\epsilon|X)$ and replace the top-k[11] non-overlapping spans with the predicted entity. Our previous example might be partially decoded as:

> **Tony_Adams_(footballer)** *and P ##latt are both injured and will miss England 's opening* **1998_FIFA_World_Cup** *qualifier ...*

In the next iteration, decoded entities (**bold**) are represented by $\mathcal{E}_{\text{E-BERT}}$ in the input, while non-decoded spans continue to be represented by $\mathcal{E}_{\text{BERT}}$ (see Figure 4). We set the maximum number of iterations to $J = 3$, as there were no improvements beyond that point on the dev set.

Baselines. We train two baselines that combine BERT and Wikipedia2Vec without vector space alignment:

Wikipedia2Vec-BERT-MLM: BERT and its pretrained MLM head, finetuned to predict non-aligned Wikipedia2Vec vectors. In practice, this means replacing $\mathcal{E}_{\text{E-BERT}}$ with $\mathcal{E}_{\text{Wikipedia}}$ in Eq. 2. Embedding the *[E-MASK]* token with non-aligned $\mathcal{E}_{\text{Wikipedia}}$ led to a drop in dev set micro F1, therefore we report this baseline with the standard *[MASK]* token.

[11] $k = \text{ceil}(\frac{j(m+n)}{J}) - m$, where $1 \leq j \leq J$ is the current iteration, m is the number of already decoded entities from previous iterations, and $n = |\{X : \text{argmax}_a[p(a|X)] \neq \epsilon\}|$.

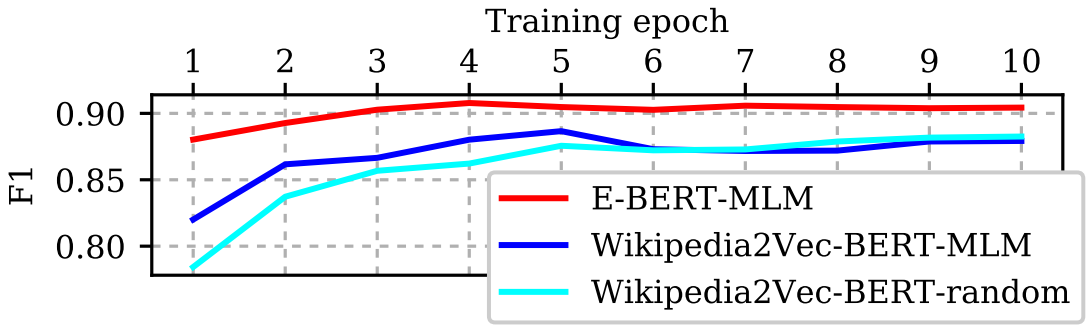

Figure 5: AIDA dev set micro F1 after every epoch.

Wikipedia2Vec-BERT-random: Like Wikipedia2Vec-BERT-MLM, but the MLM head is replaced by a randomly initialized layer.

Data. We train and evaluate on AIDA, a news dataset annotated with Wikipedia URLs (Hoffart et al., 2011). To ensure coverage of the necessary entities, we include all gold entities and all generator candidates in the entity vocabulary $\mathbb{L}_{\text{Ent}}$, even if they fall under the Wikipedia2Vec link threshold (see Section 3.3). While this is based on the unrealistic assumption that we know the contents of the test set in advance, it is necessary for comparability with Peters et al. (2019), Kolitsas et al. (2018) and Broscheit (2019), who also design their entity vocabulary around the data. See Appendix for more details on data and preprocessing. We evaluate strong match F1, i.e., a prediction must have the same start, end and entity (URL) as the gold standard. URLs that redirect to the same Wikipedia page are considered equivalent.

Hyperparameters. We train with Adam and a linear learning rate scheduler (10% warmup) for 10 epochs, and we select the best epoch on the dev set. Peak learning rate and batch size are tuned on the dev set (see Appendix).

	P	R	F1
E-BERT-MLM	21.1	61.8	31.5
w/ standard *[MASK]* token	23.3	65.2	34.3
Wikipedia2Vec-BERT-MLM	1.3	8.3	2.3
Wikipedia2Vec-BERT-random	1.3	6.8	2.2

Table 7: AIDA dev set micro precision / recall / F1 (%) before finetuning. Results without iterative refinement.

Results and discussion. Table 6 shows that E-BERT-MLM is competitive with previous work on AIDA. The aligned entity vectors play a key role in this performance, as they give the model a good initialization for predicting entities from context. When we remove this initialization by using non-aligned entity vectors (Wikipedia2Vec-BERT baselines), we get worse unsupervised performance (Table 7), slower convergence during finetuning (Figure 5), and a lower final F1 (Table 6).

6 Conclusion

We introduced **E-BERT**, an efficient yet effective way of injecting factual knowledge about entities into the BERT pretrained Language Model. We showed how to align Wikipedia2Vec entity vectors with BERT's wordpiece vector space, and how to feed the aligned vectors into BERT as if they were wordpiece vectors. In doing so, we made no changes to the BERT encoder itself. This stands in contrast to other entity-enhanced versions of BERT, such as ERNIE or KnowBert, which add encoder layers and require expensive further pretraining.

We set a new state of the art on LAMA, a recent unsupervised QA benchmark. Furthermore, we presented evidence that the original BERT model sometimes relies on the surface forms of entity names (rather than "true" factual knowledge) for this task. To quantify this effect, we introduced LAMA-UHN, a subset of LAMA where questions with helpful entity names are deleted.

We also showed how to apply E-BERT to two supervised tasks: relation classification and entity linking. On both tasks, we achieve results competitive with or better than existing baselines, but without the need for expensive pretraining.

Acknowledgements

This research was funded by Siemens AG. We thank our anonymous reviewers for their helpful comments.

References

Samuel Broscheit. 2019. Investigating entity knowledge in bert with simple neural end-to-end entity linking. In *CoNLL*, pages 677–685, Hong Kong, China.

Haotian Chen, Sahil Wadhwa, Xi David Li, and Andrej Zukov-Gregoric. 2019. YELM: End-to-end contextualized entity linking. *arXiv preprint arXiv:1911.03834*.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *ACL*, pages 358–365, Vancouver, Canada.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *EMNLP-IJCNLP*, pages 1173–1178, Hong Kong, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, USA.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show your work: Improved reporting of experimental results. In *EMNLP-IJCNLP*, pages 2185–2194, Hong Kong, China.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frédérique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *LREC*, pages 3448–3452, Miyazaki, Japan.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-Predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*, pages 6114–6123, Hong Kong, China.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*, pages 782–792, Edinburgh, UK.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2019. How can we know what language models know? *arXiv preprint arXiv:1911.12543*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *CoNLL*, pages 519–529, Brussels, Belgium.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Matthew E Peters, Mark Neumann, IV Logan, L Robert, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, Hong Kong, China.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, Hong Kong, China.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Timo Schick and Hinrich Schütze. 2019. BERTRAM: Improved word embeddings have big impact on contextualized model performance. *arXiv preprint arXiv:1910.07181*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *ICLR*, Toulon, France.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*, pages 4231–4242, Brussels, Belgium.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

Hai Wang, Dian Yu, Kai Sun, Janshu Chen, and Dong Yu. 2019a. Improving pre-trained multilingual models with vocabulary expansion. In *CoNLL*, pages 316–327, Hong Kong, China.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming

Zhou, et al. 2020. K-Adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2019b. StructBERT: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019c. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *arXiv preprint arXiv:1911.06136*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*, Berlin, Germany.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *ACL*, pages 1441–1451, Florence, Italy.

E-BERT: Efficient-Yet-Effective Entity Embeddings for BERT (Appendix)

Unsupervised QA (LAMA)

Data

We downloaded the LAMA dataset from https://dl.fbaipublicfiles.com/LAMA/data.zip. We use the LAMA-T-REx and LAMA-Google-RE relations, which are aimed at factual knowledge. Table 10 shows results on indiviual relations, as well as the number of questions per relation before and after applying the LAMA-UHN heuristics.

Preprocessing

As mentioned in Section 4.1, we do not use LAMA's oracle entity IDs. Instead, we map surface forms to entity IDs via the Wikidata query API (https://query.wikidata.org). For example, to look up *Jean Marais*:

```
SELECT ?id ?str WHERE {
  ?id rdfs:label ?str .
  VALUES ?str { 'Jean Marais'@en } .
  FILTER((LANG(?str)) = 'en') .
}
```

If more than one Wikidata ID is returned, we select the lowest one. We then map Wikidata IDs to the corresponding Wikipedia URLs:

```
SELECT ?id ?wikiurl WHERE {
  VALUES ?id { wd:Q168359 } .
  ?wikiurl schema:about ?id .
  ?wikiurl schema:inLanguage 'en' .
  FILTER REGEX(str(?wikiurl),
        '.*en.wikipedia.org.*') .
}
```

Relation classification

Data

The RC dataset, which is a subset of the FewRel corpus, was compiled by Zhang et al. (2019). We downloaded it from https://cloud.tsinghua.edu.cn/f/32668247e4fd4f9789f2/. Table 8 shows dataset statistics.

Preprocessing

The dataset contains sentences with annotated subject and object entity mentions, their oracle entity IDs and their relation (which must be predicted). We use the BERT wordpiece tokenizer to tokenize the sentence and insert special wordpieces to mark the entity mentions: # for subjects and $ for objects. Then, we insert the entity IDs. For example, an input to E-BERT-concat would look like this:

> *[CLS] Taylor was later part of the ensemble cast in MGM 's classic $* ***World_War_II*** */ World War II $ drama " #* ***Battleground_(film)*** */ Battle ##ground # " (1949) . [SEP]*

We use the oracle entity IDs of the dataset, which are also used by ERNIE (Zhang et al., 2019).

Hyperparameters

We tune peak learning rate and number of epochs on the dev set (selection criterion: macro F1). We do a full search over the same hyperparameter space as Zhang et al. (2019):

Learning rate: $[2 \cdot 10^{-5}, 3 \cdot 10^{-5}, \mathbf{5 \cdot 10^{-5}}]$

Number of epochs: $[3, 4, 5, 6, 7, 8, 9, \mathbf{10}]$

The best configuration for E-BERT-concat is marked in bold. Figure 6 shows expected maximum performance as a function of the number of evaluated configurations (Dodge et al., 2019).

Entity linking (AIDA)

Data

We downloaded the AIDA dataset from:

- https://allennlp.s3-us-west-2.amazonaws.com/knowbert/wiki_entity_linking/aida_train.txt
- https://allennlp.s3-us-west-2.amazonaws.com/knowbert/wiki_entity_linking/aida_dev.txt
- https://allennlp.s3-us-west-2.amazonaws.com/knowbert/wiki_entity_linking/aida_test.txt

Preprocessing

Each AIDA file contains documents with annotated entity spans (which must be predicted). The documents are already whitespace tokenized, and we further tokenize words into wordpieces with the standard BERT tokenizer. If a document is too long (length > 512), we split it into smaller chunks by (a) finding the sentence boundary that is closest to the document midpoint, (b) splitting the document, and (c) repeating this process recursively until all chunks are short enough. Table 9 shows dataset statistics.

Hyperparameters

We tune batch size and peak learning rate on the AIDA dev set (selection criterion: strong match micro F1). We do a full search over the following hyperparameter space:

Batch size: $[16, 32, 64, \mathbf{128}]$

Learning rate: $[\mathbf{2 \cdot 10^{-5}}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}]$

The best configuration for E-BERT-MLM is marked in bold. Figure 7 shows expected maximum performance as a function of the number of evaluated configurations (Dodge et al., 2019).

# relations	80		
# unique entities	54648		
	train	dev	test
# samples	8000	16000	16000
# samples per relation	100	200	200

Table 8: Relation classification dataset statistics.

# unique gold entities	5574		
# unique candidate entities	463663		
	train	dev	test
# documents	946	216	231
# documents (after chunking)	1111	276	271
# potential spans (candidate generator)	153103	38012	34936
# gold entities	18454	4778	4478

Table 9: Entity linking (AIDA) dataset statistics.

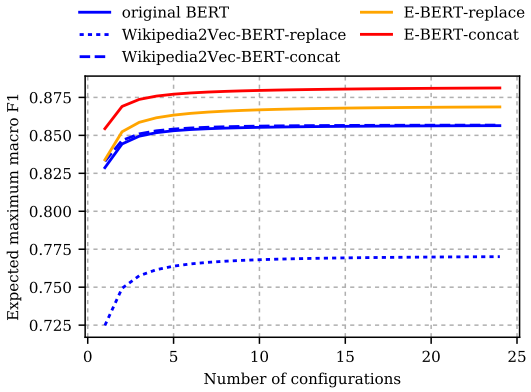

Figure 6: Relation classification: Expected maximum macro F1 (dev set) as a function of the number of hyperparameter configurations.

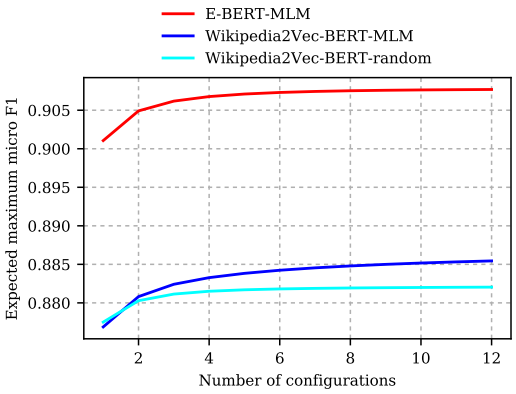

Figure 7: Entity linking: Expected maximum micro F1 (dev set) as a function of the number of hyperparameter configurations.

Relation (dataset)	Model / Model size:	original BERT	E-BERT replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	number of questions
		BASE					LARGE			
T-REx:P17	(0, original LAMA)	31.3	53.7	52.4	55.3	23.7	36.5	43.3	42.8	930
T-REx:P17	(1)	31.0	55.0	53.3	55.5	23.2	36.2	44.5	43.3	885
T-REx:P17	(2, LAMA-UHN)	31.0	55.0	53.3	55.5	23.2	36.2	44.5	43.3	885
T-REx:P19	(0, original LAMA)	21.1	26.4	28.1	28.7	23.3	22.2	24.6	25.3	944
T-REx:P19	(1)	20.6	26.5	27.5	28.2	22.9	21.8	24.5	24.8	933
T-REx:P19	(2, LAMA-UHN)	9.8	20.3	18.7	19.4	12.2	11.7	18.1	15.5	728
T-REx:P20	(0, original LAMA)	27.9	29.7	35.8	16.6	31.1	31.7	37.1	33.5	953
T-REx:P20	(1)	28.2	29.9	36.0	16.5	31.0	32.0	37.2	33.8	944
T-REx:P20	(2, LAMA-UHN)	15.5	21.5	23.3	8.4	20.0	18.9	27.3	22.6	656
T-REx:P27	(0, original LAMA)	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	966
T-REx:P27	(1)	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	945
T-REx:P27	(2, LAMA-UHN)	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.2	423
T-REx:P30	(0, original LAMA)	25.4	69.9	69.8	66.8	24.0	28.0	75.0	60.4	975
T-REx:P30	(1)	25.1	70.3	69.9	66.6	23.9	27.5	75.0	60.3	963
T-REx:P30	(2, LAMA-UHN)	25.1	70.3	69.9	66.6	23.9	27.5	75.0	60.3	963
T-REx:P31	(0, original LAMA)	36.7	25.5	46.9	43.7	18.7	30.2	12.3	16.1	922
T-REx:P31	(1)	21.1	28.4	35.8	30.3	12.4	16.3	9.9	9.8	564
T-REx:P31	(2, LAMA-UHN)	21.1	28.4	35.8	30.3	12.4	16.3	9.9	9.8	564
T-REx:P36	(0, original LAMA)	62.2	42.1	61.6	57.3	62.2	67.0	44.7	66.0	703
T-REx:P36	(1)	51.5	41.9	53.9	45.9	51.7	57.5	43.8	58.8	534
T-REx:P36	(2, LAMA-UHN)	51.5	41.9	53.9	45.9	51.7	57.5	43.8	58.8	534
T-REx:P37	(0, original LAMA)	54.6	51.2	56.5	60.2	53.1	61.5	54.3	62.7	966
T-REx:P37	(1)	52.9	51.6	55.5	59.4	51.9	60.5	54.2	62.1	924
T-REx:P37	(2, LAMA-UHN)	52.9	51.6	55.5	59.4	51.9	60.5	54.2	62.1	924
T-REx:P39	(0, original LAMA)	8.0	22.9	22.5	17.0	17.2	4.7	8.1	8.6	892
T-REx:P39	(1)	7.5	23.0	22.3	17.1	16.5	4.6	8.1	8.5	878
T-REx:P39	(2, LAMA-UHN)	7.5	23.0	22.3	17.1	16.5	4.6	8.1	8.5	878
T-REx:P47	(0, original LAMA)	13.7	8.9	10.8	9.8	14.0	18.2	15.1	15.9	922
T-REx:P47	(1)	13.6	9.1	10.7	9.6	13.9	18.6	15.2	15.9	904
T-REx:P47	(2, LAMA-UHN)	13.6	9.1	10.7	9.6	13.9	18.6	15.2	15.9	904
T-REx:P101	(0, original LAMA)	9.9	37.8	40.8	16.7	12.2	11.5	37.8	36.1	696
T-REx:P101	(1)	9.5	38.2	40.9	16.1	11.4	10.8	38.0	35.8	685
T-REx:P101	(2, LAMA-UHN)	9.5	38.2	40.9	16.1	11.4	10.8	38.0	35.8	685
T-REx:P103	(0, original LAMA)	72.2	85.8	86.8	85.5	73.4	78.2	84.4	84.9	977
T-REx:P103	(1)	72.1	85.7	86.8	85.4	73.3	78.2	84.4	84.9	975
T-REx:P103	(2, LAMA-UHN)	45.8	81.9	74.7	83.6	72.2	58.6	81.2	71.1	415
T-REx:P106	(0, original LAMA)	0.6	6.5	5.4	8.4	1.6	0.6	4.3	2.1	958
T-REx:P106	(1)	0.6	6.5	5.4	8.4	1.6	0.6	4.3	2.1	958
T-REx:P106	(2, LAMA-UHN)	0.6	6.5	5.4	8.4	1.6	0.6	4.3	2.1	958
T-REx:P108	(0, original LAMA)	6.8	9.9	23.2	14.1	10.7	1.6	11.7	15.9	383
T-REx:P108	(1)	6.5	9.9	23.0	13.9	10.5	1.3	11.8	16.0	382
T-REx:P108	(2, LAMA-UHN)	6.5	9.9	23.0	13.9	10.5	1.3	11.8	16.0	382
T-REx:P127	(0, original LAMA)	34.8	24.0	34.9	36.2	31.4	34.8	25.3	35.8	687
T-REx:P127	(1)	14.2	19.7	23.5	17.1	15.5	14.6	21.1	24.6	451
T-REx:P127	(2, LAMA-UHN)	14.2	19.7	23.5	17.1	15.5	14.6	21.1	24.6	451

Table 10: Mean Hits@1 and number of questions per LAMA relation. 0: original LAMA dataset, 1: after applying heuristic 1 (string match filter), 2: after applying both heuristics (LAMA-UHN).

Relation (dataset)	Model size: / Model	BASE original BERT	BASE E-BERT replace	BASE E-BERT-concat	BASE ERNIE	BASE Know-Bert	LARGE original BERT	LARGE E-BERT-replace	LARGE E-BERT-concat	number of questions
T-REx:P131	(0, original LAMA)	23.3	33.4	36.4	37.3	27.7	26.3	31.4	37.2	881
T-REx:P131	(1)	16.7	32.0	33.9	32.7	21.5	20.1	31.0	33.4	706
T-REx:P131	(2, LAMA-UHN)	16.7	32.0	33.9	32.7	21.5	20.1	31.0	33.4	706
T-REx:P136	(0, original LAMA)	0.8	5.2	9.1	0.6	0.6	1.3	6.9	13.1	931
T-REx:P136	(1)	0.2	5.1	8.7	0.2	0.1	0.2	6.9	12.2	913
T-REx:P136	(2, LAMA-UHN)	0.2	5.1	8.7	0.2	0.1	0.2	6.9	12.2	913
T-REx:P138	(0, original LAMA)	61.6	8.8	26.5	0.2	63.7	45.1	2.6	24.0	645
T-REx:P138	(1)	5.0	10.0	8.8	0.0	6.9	4.4	4.4	6.2	160
T-REx:P138	(2, LAMA-UHN)	5.0	10.0	8.8	0.0	6.9	4.4	4.4	6.2	160
T-REx:P140	(0, original LAMA)	0.6	0.6	1.1	0.0	0.8	0.6	1.1	0.6	473
T-REx:P140	(1)	0.4	0.6	0.9	0.0	0.6	0.4	0.9	0.4	467
T-REx:P140	(2, LAMA-UHN)	0.4	0.6	0.9	0.0	0.6	0.4	0.9	0.4	467
T-REx:P159	(0, original LAMA)	32.4	30.3	48.3	41.8	36.8	34.7	22.3	45.2	967
T-REx:P159	(1)	23.1	31.6	41.9	34.4	28.7	25.6	20.9	37.8	843
T-REx:P159	(2, LAMA-UHN)	23.1	31.6	41.9	34.4	28.7	25.6	20.9	37.8	843
T-REx:P176	(0, original LAMA)	85.6	41.6	74.6	81.8	90.0	87.5	36.6	81.3	982
T-REx:P176	(1)	31.4	42.9	51.8	26.2	51.3	40.8	44.5	57.1	191
T-REx:P176	(2, LAMA-UHN)	31.4	42.9	51.8	26.2	51.3	40.8	44.5	57.1	191
T-REx:P178	(0, original LAMA)	62.8	49.8	66.6	60.1	70.3	70.8	51.2	69.4	592
T-REx:P178	(1)	40.7	42.6	51.6	36.9	52.2	53.6	51.1	57.7	366
T-REx:P178	(2, LAMA-UHN)	40.7	42.6	51.6	36.9	52.2	53.6	51.1	57.7	366
T-REx:P190	(0, original LAMA)	2.4	2.9	2.5	2.6	2.8	2.3	2.3	2.8	995
T-REx:P190	(1)	1.5	2.4	1.6	1.6	2.0	1.7	1.9	2.3	981
T-REx:P190	(2, LAMA-UHN)	1.5	2.4	1.6	1.6	2.0	1.7	1.9	2.3	981
T-REx:P264	(0, original LAMA)	9.6	30.5	33.6	13.3	21.2	8.2	23.1	15.6	429
T-REx:P264	(1)	9.6	30.6	33.4	13.3	21.3	8.2	23.1	15.7	428
T-REx:P264	(2, LAMA-UHN)	9.6	30.6	33.4	13.3	21.3	8.2	23.1	15.7	428
T-REx:P276	(0, original LAMA)	41.5	23.8	47.7	48.4	43.3	43.8	23.1	51.8	959
T-REx:P276	(1)	19.8	26.1	31.7	27.0	20.6	23.4	25.0	36.0	625
T-REx:P276	(2, LAMA-UHN)	19.8	26.1	31.7	27.0	20.6	23.4	25.0	36.0	625
T-REx:P279	(0, original LAMA)	30.7	14.7	30.7	29.4	31.6	33.5	15.5	29.8	963
T-REx:P279	(1)	3.8	8.6	8.0	4.6	5.3	6.8	8.6	10.1	474
T-REx:P279	(2, LAMA-UHN)	3.8	8.6	8.0	4.6	5.3	6.8	8.6	10.1	474
T-REx:P361	(0, original LAMA)	23.6	19.6	23.0	25.8	26.6	27.4	22.3	25.4	932
T-REx:P361	(1)	12.6	17.9	17.7	13.7	15.3	18.5	20.2	22.0	633
T-REx:P361	(2, LAMA-UHN)	12.6	17.9	17.7	13.7	15.3	18.5	20.2	22.0	633
T-REx:P364	(0, original LAMA)	44.5	61.7	64.0	48.0	40.9	51.1	60.6	61.3	856
T-REx:P364	(1)	43.5	61.7	63.5	47.4	40.0	50.7	60.5	61.2	841
T-REx:P364	(2, LAMA-UHN)	43.5	61.7	63.5	47.4	40.0	50.7	60.5	61.2	841
T-REx:P407	(0, original LAMA)	59.2	68.0	68.8	53.8	60.1	62.1	57.9	56.3	877
T-REx:P407	(1)	57.6	69.5	67.9	53.1	58.6	61.0	59.0	55.2	834
T-REx:P407	(2, LAMA-UHN)	57.6	69.5	67.9	53.1	58.6	61.0	59.0	55.2	834
T-REx:P413	(0, original LAMA)	0.5	0.1	0.0	0.0	41.7	4.1	14.0	7.0	952
T-REx:P413	(1)	0.5	0.1	0.0	0.0	41.7	4.1	14.0	7.0	952
T-REx:P413	(2, LAMA-UHN)	0.5	0.1	0.0	0.0	41.7	4.1	14.0	7.0	952

Table 11: Mean Hits@1 and number of questions per LAMA relation (cont'd). 0: original LAMA dataset, 1: after applying heuristic 1 (string match filter), 2: after applying both heuristics (LAMA-UHN).

	Model size:	BASE					LARGE			
Relation (dataset	Model	original BERT	E-BERT replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	number of questions
T-REx:P449	(0, original LAMA)	20.9	30.9	34.7	33.8	57.0	24.0	32.5	28.6	881
T-REx:P449	(1)	18.8	31.1	33.4	32.0	56.0	21.8	32.9	27.5	848
T-REx:P449	(2, LAMA-UHN)	18.8	31.1	33.4	32.0	56.0	21.8	32.9	27.5	848
T-REx:P463	(0, original LAMA)	67.1	61.8	68.9	43.1	35.6	61.3	52.0	66.7	225
T-REx:P463	(1)	67.1	61.8	68.9	43.1	35.6	61.3	52.0	66.7	225
T-REx:P463	(2, LAMA-UHN)	67.1	61.8	68.9	43.1	35.6	61.3	52.0	66.7	225
T-REx:P495	(0, original LAMA)	16.5	46.3	48.3	1.0	30.8	29.7	56.7	46.9	909
T-REx:P495	(1)	15.0	46.0	47.5	0.9	29.6	28.5	56.6	46.2	892
T-REx:P495	(2, LAMA-UHN)	15.0	46.0	47.5	0.9	29.6	28.5	56.6	46.2	892
T-REx:P527	(0, original LAMA)	11.1	7.4	11.9	5.4	12.9	10.5	8.9	12.9	976
T-REx:P527	(1)	5.7	7.6	8.7	0.5	3.0	4.2	8.7	6.3	804
T-REx:P527	(2, LAMA-UHN)	5.7	7.6	8.7	0.5	3.0	4.2	8.7	6.3	804
T-REx:P530	(0, original LAMA)	2.8	1.8	2.0	2.3	2.8	2.7	2.3	2.8	996
T-REx:P530	(1)	2.8	1.8	2.0	2.3	2.8	2.7	2.3	2.8	996
T-REx:P530	(2, LAMA-UHN)	2.8	1.8	2.0	2.3	2.8	2.7	2.3	2.8	996
T-REx:P740	(0, original LAMA)	7.6	10.5	14.7	0.0	10.4	6.0	13.1	10.4	936
T-REx:P740	(1)	5.9	10.3	13.5	0.0	9.0	5.2	12.7	9.5	910
T-REx:P740	(2, LAMA-UHN)	5.9	10.3	13.5	0.0	9.0	5.2	12.7	9.5	910
T-REx:P937	(0, original LAMA)	29.8	33.0	38.8	40.0	32.3	24.9	28.3	34.5	954
T-REx:P937	(1)	29.9	32.9	38.7	39.9	32.2	24.8	28.2	34.4	950
T-REx:P937	(2, LAMA-UHN)	29.9	32.9	38.7	39.9	32.2	24.8	28.2	34.4	950
T-REx:P1001	(0, original LAMA)	70.5	56.9	76.0	75.7	73.0	73.3	49.5	78.0	701
T-REx:P1001	(1)	38.1	67.7	66.7	65.6	43.4	40.7	60.3	66.7	189
T-REx:P1001	(2, LAMA-UHN)	38.1	67.7	66.7	65.6	43.4	40.7	60.3	66.7	189
T-REx:P1303	(0, original LAMA)	7.6	20.3	26.6	5.3	9.1	12.5	29.7	33.2	949
T-REx:P1303	(1)	7.6	20.3	26.6	5.3	9.1	12.5	29.7	33.2	949
T-REx:P1303	(2, LAMA-UHN)	7.6	20.3	26.6	5.3	9.1	12.5	29.7	33.2	949
T-REx:P1376	(0, original LAMA)	73.9	41.5	62.0	71.8	75.2	82.1	47.4	70.1	234
T-REx:P1376	(1)	74.8	42.2	62.8	73.4	75.2	83.5	48.6	72.0	218
T-REx:P1376	(2, LAMA-UHN)	74.8	42.2	62.8	73.4	75.2	83.5	48.6	72.0	218
T-REx:P1412	(0, original LAMA)	65.0	54.0	67.8	73.1	69.2	63.6	49.3	61.2	969
T-REx:P1412	(1)	65.0	54.0	67.8	73.1	69.2	63.6	49.3	61.2	969
T-REx:P1412	(2, LAMA-UHN)	37.7	42.9	47.4	69.2	65.7	51.5	43.5	54.8	361
Google-RE:date_of_birth	(0)	1.6	1.5	1.9	1.9	2.4	1.5	1.5	1.3	1825
Google-RE:date_of_birth	(1)	1.6	1.5	1.9	1.9	2.4	1.5	1.5	1.3	1825
Google-RE:date_of_birth	(2)	1.6	1.5	1.9	1.9	2.4	1.5	1.5	1.3	1825
Google-RE:place_of_birth	(0)	14.9	16.2	16.9	17.7	17.4	16.1	14.8	16.6	2937
Google-RE:place_of_birth	(1)	14.9	16.2	16.8	17.7	17.4	16.0	14.8	16.6	2934
Google-RE:place_of_birth	(2)	5.9	9.4	8.2	10.3	9.4	7.2	8.5	7.9	2451
Google-RE:place_of_death	(0)	13.1	12.8	14.9	6.4	13.4	14.0	17.0	14.9	766
Google-RE:place_of_death	(1)	13.1	12.8	14.9	6.4	13.4	14.0	17.0	14.9	766
Google-RE:place_of_death	(2)	6.6	7.5	7.8	2.0	7.5	7.6	11.8	8.9	655

Table 12: Mean Hits@1 and number of questions per LAMA relation (cont'd). 0: original LAMA dataset, 1: after applying heuristic 1 (string match filter), 2: after applying both heuristics (LAMA-UHN).

Chapter 4

Corresponds to the following publication:

> Nina Poerner and Hinrich Schütze. 2019. Multi-view domain adapted sentence embeddings for low-resource unsupervised duplicate question detection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1630–1641, Hong Kong, China

Declaration of Co-Authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission.

Notes and corrections: Table 1 of the publication contains an error: The order of the rows (contextualized vs. non-contextualized) should be reversed.

[1] `https://www.aclweb.org/anthology/faq/copyright/` [accessed 05/03/2021]

Multi-View Domain Adapted Sentence Embeddings for Low-Resource Unsupervised Duplicate Question Detection

Nina Poerner[1,2] **and Hinrich Schütze**[1]
[1]Center for Information and Language Processing, LMU Munich, Germany
[2]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

Abstract

We address the problem of Duplicate Question Detection (DQD) in low-resource domain-specific Community Question Answering forums. Our multi-view framework MV-DASE combines an ensemble of sentence encoders via Generalized Canonical Correlation Analysis, using unlabeled data only. In our experiments, the ensemble includes generic and domain-specific averaged word embeddings, domain-finetuned BERT and the Universal Sentence Encoder. We evaluate MV-DASE on the CQADupStack corpus and on additional low-resource Stack Exchange forums. Combining the strengths of different encoders, we significantly outperform BM25, all single-view systems as well as a recent supervised domain-adversarial DQD method.

1 Introduction

Duplicate Question Detection is the task of finding questions in a database that are equivalent to an incoming query. Many Community Question Answering (CQA) forums leave this task to the collective memory of their users. This results in unnecessary manual work for community members as well as delayed access to answers (Hoogeveen et al., 2015).

Automatic DQD is often approached as a supervised problem with community-generated training labels. However, smaller CQA forums may suffer from label sparsity: On Stack Exchange, 50% of forums have fewer than 160 user-labeled duplicates, and 25% have fewer than 50 (see Figure 1).[1]

To overcome this problem, two avenues have been explored: The first is supervised domain-adversarial training on a label-rich source forum (Shah et al., 2018), which works best when source and target domains are related. The second is unsupervised DQD via representation learning (Charlet and Damnati, 2017; Lau and Baldwin, 2016), which requires only unlabeled questions. In this paper, we take the unsupervised avenue.

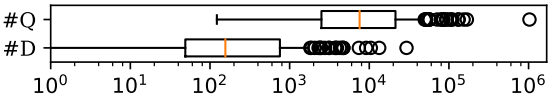

Figure 1: Distribution (log-scale box plot) of number of questions (#Q) and number of labeled duplicates (#D) on Stack Exchange. $N = 165$ forums.

A major challenge in the context of domain-specific CQA forums is that language usage may differ from the "generic" domains of existing representations. To illustrate this point, compare the following Nearest Neighbor lists of the word "tree", based either on generic GloVe embeddings (Pennington et al., 2014) or on FastText embeddings (Bojanowski et al., 2017) that were trained on specific CQA forums:

generic (GloVe): trees, branches, leaf
chess: searches, prune, modify
outdoors: trees, trunk, trunks
gis: strtree, rtree, btree
wordpress: trees, hierachy, hierarchial
gaming: trees, treehouse, skills

Charlet and Damnati (2017) and Lau and Baldwin (2016) report that representations trained on in-domain data perform better on unsupervised DQD than generic representations. But in a low-resource setting, the amount of unlabeled in-domain data is limited. This can result in low coverage or quality, as illustrated by the in-domain embedding neighbors of "tree" in the smallest forum from our dataset:

windowsphone: dreamspark, l535ds, generally

[1] archive.org/details/stackexchange [data dump: 2018-12-20]

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1630–1641,
Hong Kong, China, November 3–7, 2019.

	generic	domain-specific
contextualized	f_G: GloVe	f_D: FastText (in-domain)
noncontextualized	f_U: USE	f_B: BERT (domain-finetuned)

Table 1: Ensemble used in our experiments.

It is therefore desirable to combine the overall quality and coverage of generic representations with the domain-specificity of in-domain representations via multi-view learning. There is a large body of work on multi-view word embeddings (see Section 2.3), including domain adapted word embeddings (Sarma et al., 2018).

Recent representation learning techniques go beyond the word level and embed larger contexts (e.g., sentences) jointly (Peters et al., 2018; Devlin et al., 2019; Cer et al., 2018). To reflect this paradigm shift, we take multi-view representation learning from the word to the sentence level and propose MV-DASE (Multi-View Domain Adapted Sentence Embeddings), a framework that combines an ensemble of sentence encoders via Generalized Canonical Correlation Analysis (see Section 3.1).

MV-DASE uses **unlabeled in-domain data only**, making it applicable to the problem of unsupervised DQD. As a framework, it is **agnostic** to the internal specifics of its ensemble. In Section 3.2, we describe an ensemble of different sentence encoders: domain-specific and generic, contextualized and noncontextualized (see Table 1). In Sections 4 and 5, we demonstrate that MV-DASE is effective at **duplicate retrieval** on the CQADupStack corpus (Hoogeveen et al., 2015) and on additional low-resource Stack Exchange forums. **Significance tests** show significant gains over BM25, all single-view systems and domain-adversarial supervised training as proposed by Shah et al. (2018). In Sections 6 and 7, we successfully evaluate MV-DASE on **two additional benchmarks**: the SemEval-2017 DQD shared task (Nakov et al., 2017) as well as the unsupervised STS Benchmark (Cer et al., 2017).

2 Related Work

2.1 Duplicate Question Detection

Most prior work on DQD (e.g., Bogdanova et al. (2015); Dos Santos et al. (2015); Baldwin et al. (2016); Zhang et al. (2017); Rodrigues et al. (2017); Hoogeveen et al. (2018)) focuses on supervised architectures. As discussed, these approaches are not applicable to forums with few or no labeled duplicates.

Shah et al. (2018) tackle label sparsity by domain-adversarial training (ADA). More specifically, they train a bidirectional Long-Short Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997) on a label-rich source forum, while minimizing the distance between source and target domain representations. Their approach beats BM25 and a simple transfer baseline in cases where source and target domain are closely related (e.g., *AskUbuntu→SuperUser*), but not on more distant pairings. This is not ideal, as the existence of a big related source forum is not guaranteed.

An alternative is unsupervised DQD via representation learning, which does not require any labels. Charlet and Damnati (2017) use a word embedding-based soft cosine distance for duplicate ranking. In a recent DQD shared task (SemEval-2017 task 3B, Nakov et al. (2017)), their best unsupervised system trails the best supervised system by only 2% Mean Average Precision (MAP). This seems reasonable, given that the implicit objective of many representation learning methods (similar representations for similar objects) is closely related to the notion of a duplicate.

Charlet and Damnati (2017) report overall better results when embeddings are trained on domain-specific data rather than Wikipedia. However, they make no attempts to combine the two domains. Lau and Baldwin (2016) evaluate two representation learning techniques (doc2vec (Le and Mikolov, 2014) and word2vec (Mikolov et al., 2013a)) on CQADupStack. They also report better results when representations are learned on domain-specific rather than generic data.

2.2 Sentence embeddings and STS

Unsupervised DQD is related to the task of unsupervised Semantic Textual Similarity (STS), i.e., sentence similarity scoring (Cer et al., 2017). Arora et al. (2017) show that a weighted average over pre-trained word embeddings, followed by principal component removal, is a strong baseline for STS. We use their weighting scheme, Smooth Inverse Frequency (SIF), in Section 3.2.

Averaged word embeddings are insensitive to word order. This stands in contrast to contextualized encoders, such as LSTMs or Transform-

ers (Vaswani et al., 2017). Contextualized encoders are typically trained as unsupervised language models (Peters et al., 2018; Devlin et al., 2019) or on supervised transfer tasks (Conneau et al., 2017; Cer et al., 2018). At the time of writing, weighted averaged word embeddings achieve better results than contextualized encoders on unsupervised STS.[2]

2.3 Multi-view word embeddings

Multi-view representation learning is an umbrella term for methods that transform different representations of the same entities into a common space. In NLP, it has typically been applied to word embeddings. A famous example is the cross-lingual projection of word embeddings (Mikolov et al., 2013b; Faruqui and Dyer, 2014). Monolingually, Rastogi et al. (2015) use Generalized Canonical Correlation Analysis (GCCA) to project different word representations into a common space. Yin and Schütze (2016) combine word embeddings by concatenation, truncated Singular Value Decomposition and linear projections; Bollegala and Bao (2018) use autoencoders. Sarma et al. (2018) correlate generic and domain-specific word embeddings by Canonical Correlation Analysis (CCA).

All of these methods are post-training, i.e., they are applied to fully trained word embeddings. MV-DASE falls into the same category, albeit at the sentence level (see Section 3.1). Other methods, which we will call in-training, encourage the alignment of embeddings during training (e.g., Bollegala et al. (2015); Yang et al. (2017)).

2.4 Multi-view sentence embeddings

Multi-view sentence embeddings are less frequently explored than multi-view word embeddings. One exception is Tang and de Sa (2019), who train a recurrent neural network and an average word embedding encoder jointly on an unlabeled corpus. This method is in-training, i.e., it cannot be used to combine pre-existing encoders.

Kiela et al. (2018) dynamically integrate an ensemble of word embeddings into a task-specific LSTM. They require labeled data and the resulting embeddings are task-specific.

Sarma et al. (2018) marry domain-adapted word embeddings (see Section 2.3) with InferSent (Conneau et al., 2017), a bidirectional LSTM sentence encoder trained on Stanford Natural Language Inference (SNLI) (Bowman et al., 2015). They initialize InferSent with the adapted embeddings and then retrain it on SNLI. Note that this approach is not feasible when the training regime of an encoder cannot be reproduced, e.g., when the original training data is not publicly available.

[2] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

3 Method

We now describe MV-DASE as a general framework. For details on the ensemble used in this paper, see Section 3.2.

3.1 Framework

GCCA basics. Given zero-mean random vectors $\mathbf{x}_1 \in \mathbb{R}^{d_1}, \mathbf{x}_2 \in \mathbb{R}^{d_2}$, Canonical Correlation Analysis (CCA) finds linear transformations $\boldsymbol{\theta}_1 \in \mathbb{R}^{d_1}, \boldsymbol{\theta}_2 \in \mathbb{R}^{d_2}$ such that $\boldsymbol{\theta}_1^T \mathbf{x}_1$ and $\boldsymbol{\theta}_2^T \mathbf{x}_2$ are maximally correlated. Bach and Jordan (2002) show that CCA reduces to a generalized eigenvalue problem. A generalized eigenvalue problem finds scalar-vector pairs $(\rho, \boldsymbol{\theta})$ that satisfy $\mathbf{A}\boldsymbol{\theta} = \rho\mathbf{B}\boldsymbol{\theta}$ for matrices $\mathbf{A}, \mathbf{B}$. Here, $\mathbf{A}, \mathbf{B}$ are the following block matrices:

$$\begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \mathbf{0} \end{bmatrix} \boldsymbol{\theta} = \rho \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{2,2} \end{bmatrix} \boldsymbol{\theta} \quad (1)$$

where $\boldsymbol{\Sigma}_{1,1}, \boldsymbol{\Sigma}_{2,2}$ are the covariance matrices of $\mathbf{x}_1, \mathbf{x}_2$ and $\boldsymbol{\Sigma}_{1,2}, \boldsymbol{\Sigma}_{2,1}$ are their cross-covariance matrices. We stack all d eigenvectors into an operator $\boldsymbol{\Theta} \in \mathbb{R}^{d \times d_1 + d_2}$. Using this operator, multi-view representations are projected as:

$$\mathbf{x}_{\text{mv}} = \boldsymbol{\Theta} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \quad (2)$$

Generalized CCA (GCCA) generalizes CCA to three or more random vectors $\mathbf{x}_1 \ldots \mathbf{x}_J$. There are several variants of GCCA (Kettenring, 1971); we follow Bach and Jordan (2002) and solve a multi-view version of Equation 1:

$$\begin{aligned} &\begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{\ldots} & \boldsymbol{\Sigma}_{1,J} \\ \boldsymbol{\Sigma}_{\ldots} & \mathbf{0} & \boldsymbol{\Sigma}_{\ldots} \\ \boldsymbol{\Sigma}_{J,1} & \boldsymbol{\Sigma}_{\ldots} & \mathbf{0} \end{bmatrix} \boldsymbol{\theta} \\ &= \rho \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\ldots} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_{J,J} \end{bmatrix} \boldsymbol{\theta} \end{aligned} \quad (3)$$

For stability, we add $\tau\sigma_j \mathbf{I}_j$ to every covariance matrix $\boldsymbol{\Sigma}_{j,j}$, where τ is a hyperparameter (here: $\tau = 0.1$), $\mathbf{I}_j$ is the identity matrix and σ_j is the

average variance of $\mathbf{x}_j$. Like in the two-view case, we stack all d eigenvectors into an operator: $\mathbf{\Theta} \in \mathbb{R}^{d \times \sum_j d_j}$.

GCCA application. Assume that we have an ensemble of J sentence encoders. The j'th encoder is denoted $f_j : \mathbb{S} \rightarrow \mathbb{R}^{d_j}$, where $\mathbb{S}$ is the set of all possible in-domain strings (here: in-domain questions) and d_j is determined by f_j. Assume also that we have a sample from $\mathbb{S}$, i.e., a corpus of unlabeled in-domain strings, denoted $S = \{s_1, \ldots, s_N\}$. From this corpus, we create one training matrix $\mathbf{X}_j$ per encoder:

$$\mathbf{X}_j \in \mathbb{R}^{N \times d_j} = \begin{bmatrix} — & f_j(s_1) & — \\ & \vdots & \\ — & f_j(s_N) & — \end{bmatrix} \quad (4)$$

From $\mathbf{X}_j$ we estimate mean vector $\bar{\mathbf{x}}_j \in \mathbb{R}^{d_j}$, covariance matrix $\mathbf{\Sigma}_{j,j} \in \mathbb{R}^{d_j \times d_j}$ and cross-covariance matrices $\mathbf{\Sigma}_{j,j'} \in \mathbb{R}^{d_j \times d_{j'}}$. We then apply GCCA as described before, yielding $\mathbf{\Theta} \in \mathbb{R}^{d \times \sum_j d_j}$. The multi-view embedding of a new input q (e.g., a test query) is:

$$f_{\text{mv}}(q) = \mathbf{\Theta} \begin{bmatrix} f_1(q) - \bar{\mathbf{x}}_1 \\ \vdots \\ f_J(q) - \bar{\mathbf{x}}_J \end{bmatrix} \quad (5)$$

3.2 Ensemble

We use MV-DASE on the following ensemble:

- weighted averaged generic GloVe vectors (Pennington et al., 2014)
- weighted averaged domain-specific FastText vectors (Bojanowski et al., 2017)
- Universal Sentence Encoder (USE) (Cer et al., 2018)
- domain-finetuned BERT (Devlin et al., 2019)

In this section, we describe the encoders in detail. Note that the choice of encoders is orthogonal to the framework and other resources could be used. Where possible, we base our selection on the literature: We choose USE over InferSent due to better performance on STS (Perone et al., 2018), and BERT over ELMo (Peters et al., 2018) due to better performance on linguistic probing tasks (Liu et al., 2019a). The choice of GloVe for generic word embeddings is based on Sarma et al. (2018).

Weighted averaged word embeddings. We denote generic and domain-specific word embeddings of some word type i as $\mathbf{w}_{G,i} \in \mathbb{R}^{d_G}$ and $\mathbf{w}_{D,i} \in \mathbb{R}^{d_D}$. For $\mathbf{w}_{G,i}$, we use pre-trained 300-d GloVe vectors.[3] $\mathbf{w}_{D,i}$ are trained using skipgram FastText[4] (100-d, default parameters) on the in-domain corpus S. We SIF-weight all word embeddings by $a \cdot (a + p(i))^{-1}$, where $p(i)$ is the unigram probability of the word type and the smoothing factor (here: $a = 10^{-3}$) is taken from Arora et al. (2017). We find that probabilities estimated on S produce better results than the Wikipedia-based probabilities provided by Arora et al. (2017) (see Table 2, top), hence this is what we use below. After weighting, we perform top-3 principal component removal on the embedding matrices, which is beneficial for word-level similarity tasks (Mu et al., 2018). We denote the new embeddings of word type i as $\hat{\mathbf{w}}_{G,i}, \hat{\mathbf{w}}_{D,i}$. The embedding of a tokenized string $s = (s_1, \ldots, s_T)$ is computed by averaging:

$$f_G(s) = \frac{1}{T} \sum_{t=1}^{T} \hat{\mathbf{w}}_{G,s_t} \quad f_D(s) = \frac{1}{T} \sum_{t=1}^{T} \hat{\mathbf{w}}_{D,s_t}$$

Contextualized encoders. USE and BERT are downloaded as pre-trained models.[5] [6] USE is a Transformer trained on SkipThought (Kiros et al., 2015), conversation response prediction (Henderson et al., 2017) and SNLI. It outputs a single 512-d sentence embedding, which we use as-is. Below, USE is denoted f_U.

	f_G (GloVe)	f_D (FastText)	f_B (BERT)
no SIF	.089	.083	.134
wiki SIF	.128	.100	.159
in-domain SIF	.147	.104	.176

	f_B (BERT)	ELMo
generic	.138	.103
domain-finetuned	.176	.155

Table 2: Mean Average Precision (MAP) averaged over heldout forums. Top: MAP as a function of whether and where SIF weights are estimated. Bottom: MAP of generic vs. domain-finetuned BERT and ELMo. Evaluation setup is as described in Section 4, using four heldout forums. Gray: best in column.

[3] nlp.stanford.edu/data/glove.42B.300d.zip

[4] github.com/facebookresearch/fastText

[5] tfhub.dev/google/universal-sentence-encoder-large/3

[6] tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1

BERT is a Transformer that was pre-trained as a masked language model with next sentence prediction. We find that domain-finetuning BERT on S results in improvements over generic BERT (see Table 2, bottom). Note that domain-finetuning refers to unsupervised training as a masked language model, i.e., we only require unlabeled data (Howard and Ruder, 2018). We use default parameters[7] except for a reduced batch size of 8.

At test time, we take the following approach: BERT segments a token sequence $s = (s_1, \ldots, s_T)$ into a subword sequence $s' = ([\text{CLS}], s'_1, \ldots, s'_{T'}, [\text{SEP}])$, where [CLS] and [SEP] are special tokens that were used during pre-training, and $T' \geq T$. BERT produces one 768-d vector $\mathbf{v}_{l,t}$ per subword s'_t and layer $l \in [1, \ldots, L]$, where L is the total number of layers (here: 12). We SIF-weight all vectors according to the probability of their subword (estimated on S) and average over layers and subwords, excluding the special tokens:

$$f_B(s) = \frac{1}{T' \cdot L} \sum_{t=1}^{T'} \sum_{l=1}^{L} \frac{a}{a + p(s'_t)} \mathbf{v}_{l,t}$$

4 Evaluation on Stack Exchange

4.1 Data

Corpora. We evaluate MV-DASE on the CQADupStack corpus (Hoogeveen et al., 2015), which is based on a 2014 Stack Exchange dump. CQADupStack contains 12 forums that have enough duplicates for supervised training; as a consequence, it may not be representative of low-resource domains. We therefore supplement it with 12 low-resource forums from the 2018-12-20 Stack Exchange dump.[8] For our purposes, low-resource means a forum with 100–200 duplicates, which we consider sufficient for evaluation but insufficient for supervised training. All duplicates in the datasets were labeled by unpaid community members. As a result, false negatives (i.e., unflagged duplicates) are common in the gold standard (Hoogeveen et al., 2016). While we do not explicitly filter for language, the vast majority of the data is in English.

[7] github.com/google-research/bert/blob/master/run_pretraining.py

[8] Preprocessed low-resource data can be downloaded here: github.com/npoe/lowresourcecqa

		forum	#Q	#D	#T
CQADupStack forums	and	android	23697	1579	2.4M
	eng	english	41791	3506	3.4M
	gam	gaming	46896	2207	4.0M
	gis	gis	38522	1099	4.6M
	mat	mathematica	17509	1271	2.6M
	phy	physics	39355	1769	6.1M
	prg	programmers	33052	1538	5.6M
	sta	stats	42921	890	7.2M
	tex	tex	71090	4939	7.4M
	uni	unix	48454	1648	5.5M
	web	webmasters	17911	1143	2.0M
	wor	wordpress	49146	719	5.6M
low-resource forums	bud	buddhism	5350	120	670K
	che	chess	4539	154	500K
	cog	cogsci	5687	126	800K
	law	law	11059	126	1.7M
	net	networkengineering	11386	154	1.5M
	out	outdoors	4651	124	580K
	pro	productivity	2508	127	380K
	rev	reverseengineering	15619	119	790K
	sit	sitecore	5605	130	680K
	spo	sports	4531	127	430K
	sqa	sqa	8360	166	950K
	win	windowsphone	3490	192	290K

Table 3: Forum statistics. #Q: total number of questions, #D: number of labeled duplicates, #T: number of tokens in training set S. Gray: heldout forums.

Data split. We split every forum into a test and training set, such that the test set contains all duplicates and the training set contains the remaining unlabeled questions.[9] The unlabeled training set is used for FastText training, BERT domain-finetuning, SIF weight estimation and GCCA. Test queries are never seen during training, not even in an unsupervised way. For hyperparameter choices, we hold out two high-resource and two low-resource forums (highlighted in Table 3). They are not used for the final evaluation and significance tests.

Preprocessing. Every question object consists of a title (average length 9 words), a body (average length 125 words), any number of answers or comments, and metadata (e.g., upvotes, view counts). We preprocess the data with the CQADupStack package.[10] To calculate question representations, we use the concatenation of question title and body. We always ignore answers, comments and metadata, as this information is not usually available at the time a question is posted.

[9] We do not use the official CQADupStack train / test split, as it is meant for supervised training and has comparatively few duplicates per test set. Since MV-DASE is unsupervised, we can afford a more robust evaluation on all labeled duplicates.

[10] github.com/D1Doris/CQADupStack

		heldout		test forums										MAP	AUC(.05)	NDCG	P@3	R@3
		and	eng	gam	gis	mat	prg	phy	sta	tex	uni	web	wor	average over test forums				
CQADupStack forums	1 BM25	.175	.162	.310	.264	.132	.119	.216	.212	.116	.171	.103	.171	.181	.821	.314	.067	.196
	2 f_G (GloVe)	.121	.093	.202	.148	.056	.084	.152	.153	.063	.120	.085	.093	.115	.755	.233	.042	.123
	3 f_D (FastText)	.123	.083	.211	.169	.079	.091	.172	.175	.085	.136	.084	.107	.131	.817	.261	.047	.138
	4 f_U (USE)	.183	.113	.347	.156	.081	.146	.195	.165	.071	.142	.110	.117	.153	.832	.285	.056	.163
	5 f_B (BERT)	.141	.129	.262	.196	.103	.099	.190	.179	.090	.135	.109	.134	.150	.805	.276	.055	.159
	6 MV-DASE	.211	.177	.371	.274	.149	.181	.259	.236	.135	.206	.145	.183	.214	.904	.362	.080	.232
	7 InferSent	.069	.047	.145	.123	.041	.041	.105	.121	.042	.078	.053	.072	.082	.667	.182	.029	.085
	8 doc2vec	.102	.057	.141	.150	.064	.069	.138	.170	.067	.125	.083	.111	.112	.799	.234	.040	.116
	9 ELMo	.141	.116	.251	.179	.081	.097	.184	.182	.087	.147	.097	.117	.142	.835	.274	.051	.149
	10 word-level CCA	.149	.109	.253	.202	.101	.111	.190	.189	.096	.156	.103	.125	.153	.851	.290	.055	.161
	11 upper bound	1.00															.351	.999
		bud	che	cog	law	net	out	pro	rev	sit	spo	sqa	win	average over test forums				
low-resource forums	12 BM25	.276	.195	.269	.345	.167	.373	.196	.186	.430	.465	.275	.349	.306	.842	.461	.116	.345
	13 f_G (GloVe)	.249	.125	.209	.312	.103	.260	.110	.134	.237	.363	.166	.239	.213	.781	.359	.079	.234
	14 f_D (FastText)	.142	.064	.132	.255	.111	.168	.067	.101	.243	.239	.173	.136	.163	.767	.314	.060	.180
	15 f_U (USE)	.332	.247	.384	.458	.152	.513	.214	.144	.282	.448	.221	.244	.306	.880	.470	.119	.352
	16 f_B (BERT)	.261	.173	.221	.335	.137	.348	.171	.143	.324	.489	.194	.257	.262	.812	.411	.099	.294
	17 MV-DASE	.378	.259	.384	.447	.184	.495	.233	.241	.427	.523	.289	.352	.358	.924	.524	.137	.407
	18 InferSent	.154	.073	.117	.236	.079	.194	.089	.078	.192	.312	.123	.161	.158	.701	.281	.054	.162
	19 doc2vec	.133	.058	.117	.239	.146	.145	.057	.080	.192	.141	.140	.090	.135	.759	.279	.048	.143
	20 ELMo	.222	.140	.228	.332	.137	.248	.136	.092	.247	.433	.171	.278	.230	.837	.387	.084	.252
	21 word-level CCA	.260	.142	.237	.325	.146	.274	.111	.186	.312	.327	.218	.194	.233	.844	.391	.086	.254
	22 ADA	.229	.164	.161	.250	.132	.207	.117	.147	.225	.299	.193	.218	.195	.823	.347	.068	.201
	23 upper bound	1.00															.341	.999

Table 4: Main results. Left: MAP on individual forums (heldout and test forums). Rightmost five columns: all metrics averaged over test forums (excluding heldout forums). Gray: best in column.

4.2 Evaluation and Metrics

Given a test query q, we rank all candidates $c \neq q$ from the same forum by $\cos(f(q), f(c))$, where f is an encoder (e.g., MV-DASE). Our metrics are MAP, AUC(.05), Normalized Discounted Cumulative Gain (NDCG), Recall@3 (R@3) and Precision@3 (P@3). AUC(.05), the area under the ROC curve up to a false positive rate of .05, is used by Shah et al. (2018). Note that upper bounds on P@3 and R@3 are not 1, since most duplicates have only one original and a few have more than three.

4.3 Baselines

Unsupervised. Our IR baseline is BM25 (Robertson et al., 1995) as implemented in Elasticsearch 6.5.4 (Gormley and Tong, 2015) with default parameters. We test against all single-view encoders from our ensemble. The remaining unsupervised baselines are:

- ELMo (Peters et al., 2018).[11] We treat ELMo like BERT (Section 3.2), i.e., we finetune[12] the language model on the in-domain corpus (3 epochs, batch size 8), SIF-weight all vectors according to in-domain word probability and then average over layers and tokens.
- Doc2vec (Le and Mikolov, 2014) trained on the in-domain corpus, using the best DQD hyperparameters reported in Lau and Baldwin (2016).
- InferSent V.1.[13] (Conneau et al., 2017)
- Our re-implementation of domain-adapted CCA word embeddings (Sarma et al. (2018), see Section 2.3). We use the same word embeddings, SIF weights and component removal described in Section 3.2. Denoted "word-level CCA" below.

[11] tfhub.dev/google/elmo/2

[12] github.com/allenai/bilm-tf/blob/master/bin/restart.py

[13] github.com/facebookresearch/InferSent

	MAP	AUC(.05)
1	MV-DASE, $\neg f_G$, $\neg f_D$	MV-DASE
2	$\neg f_B$, $\neg(f_G, f_D)$	$\neg f_B$, $\neg f_D$, $\neg f_G$, $\neg(f_G, f_D)$
3	BM25, avg, concat, ADA, word-level CCA, ELMo, $\neg f_U$, f_D, f_G, f_B, $\neg(f_B, f_U)$, f_U	BM25, avg, concat, doc2vec, ADA, word-level CCA, ELMo, f_B, f_D, f_G, $\neg f_U$, f_U, $\neg(f_B, f_U)$
4	InferSent, doc2vec	InferSent

Table 5: Group rankings by transitive closure of paired t-tests. $\neg f_j$ is MV-DASE without f_j (see Table 6). No particular order inside groups.

ADA. We evaluate the supervised domain-adversarial method of Shah et al. (2018) (ADA) on the low-resource forums. Recall that ADA requires a related labeled source domain. To achieve this, we pair every low-resource forum (target) with the CQADupStack forum (source) with which it has the highest word trigram overlap. See supplementary material for more details and a table of all source-target mappings.[14]

4.4 Ablation studies

We perform a set of experiments where we omit views from the ensemble. We also replace GCCA with naive view concatenation or view averaging. When averaging, we pad lower-dimensional vectors (Coates and Bollegala, 2018).

4.5 Significance tests

We perform paired t-tests, using the 20 test set forums as data points.[15] We then find groups of equivalent methods by transitive closure of $a \sim b \equiv p \geq .05$. Group A being ranked higher than group B means that every method in A performs significantly better than every method in B. Two methods in the same group may differ significantly, but there exists a chain between them of methods with insignificant differences.

5 Discussion

5.1 Comparison with baselines

BM25. BM25 is a tough baseline for DQD: In terms of MAP, it is better than or comparable to every single view (see Table 5). MV-DASE on the other hand, which is built from the same views, outperforms BM25 significantly and almost consistently (19 out of 20 test forums), regardless of the metric. This underlines the usefulness of our multi-view approach.

			MAP	AUC(.05)	NDCG	P@3	R@3
CQADupStack	1	$\neg f_G$	.002	.000	.000	.000	-.001
	2	$\neg f_D$	-.002	-.008	-.004	-.001	-.002
	3	$\neg f_U$	-.030	-.032	-.037	-.012	-.035
	4	$\neg f_B$	-.010	-.006	-.011	-.003	-.008
	5	$\neg(f_U, f_B)$	-.056	-.042	-.066	-.022	-.065
	6	$\neg(f_G, f_D)$	-.012	-.016	-.017	-.005	-.015
	7	concat	-.042	-.071	-.058	-.017	-.047
	8	avg	-.046	-.075	-.064	-.018	-.051
low-resource	9	$\neg f_G$	-.008	-.005	-.010	-.006	-.016
	10	$\neg f_D$	.007	-.001	.012	.002	.006
	11	$\neg f_U$	-.058	-.050	-.063	-.023	-.067
	12	$\neg f_B$	-.017	-.006	-.014	-.010	-.028
	13	$\neg(f_U, f_B)$	-.120	-.069	-.130	-.054	-.160
	14	$\neg(f_G, f_D)$	-.010	-.005	-.007	-.005	-.014
	15	concat	-.058	-.078	-.070	-.023	-.069
	16	avg	-.067	-.081	-.080	-.028	-.082

Table 6: Ablation study. Deltas relative to MV-DASE. Metrics were averaged over test forums before calculating deltas. concat/avg are naive view concatenation and averaging. Gray: better than MV-DASE.

Single views. MV-DASE outperforms the views that make up its ensemble significantly and almost consistently. There are two exceptions (out of 20 test forums): On *law* and *outdoors*, f_U (USE) performs slightly better on its own (Table 4, row 15). Since these forums are less "technical" than most, we hypothesize that they may be less in need of domain adaptation.

Word-level CCA. The word-level CCA baseline by Sarma et al. (2018) outperforms f_G and f_D on their own (see Table 4, rows 10, 21), which validates the approach. The method is directly comparable to MV-DASE$\neg(f_U, f_B)$, i.e., MV-DASE on generic and domain-specific averaged word embeddings (see Table 6). The main differences between them are (a) the order in which CCA and averaging are performed and (b) whether the CCA "vocabulary" is composed of word types or sentences. Note that in contrast to MV-DASE, word-level CCA is incompatible with contextualized embeddings, since it requires a context-independent one-to-one mapping between word types and vectors.

[14] We also experiment with Multinomial Adversarial Networks (MAN) (Chen and Cardie, 2018), a multi-source multi-target framework that can be trained on all 24 forums jointly. Initial results were not competitive with ADA, so we do not include them here. See supplementary material for details.

[15] Ten forums for t-tests involving ADA.

ADA. Supervised domain-adversarial ADA performs significantly worse than unsupervised MV-DASE (see Table 5). It is comparable to BM25 in terms of AUC(.05) (the metric used by Shah et al. (2018)), but not in terms of MAP.

Recall that we restricted the choice of source domains to the 12 CQADupStack forums. As a consequence, some target forums were paired with non-ideal source forums (e.g., *english→buddhism*). It is possible that the baseline would have performed better with a wider choice of source domains. Nonetheless, this observation highlights a key advantage of our approach: It does not depend on the availability of a label-rich related source domain (or indeed, any labels at all).

Other baselines. InferSent performs poorly on the DQD task, which is surprising given its similarity to USE. Recall that InferSent and USE are pre-trained on sentence-level SNLI, but that the training regime of USE also contains conversation response prediction. So USE is expected to be better equipped to handle (a) multi-sentence documents and (b) forum-style language.

Doc2vec is trained on the same data as f_D, but performs significantly worse. The difference between them may be due to the ability of FastText to exploit orthography. Domain-finetuned ELMo performs comparably to domain-finetuned BERT on some forums but not consistently.

5.2 Ablation study

View ablation. On the low-resource forums, omitting f_D has a beneficial effect (Table 6, row 10). This suggests that the in-domain FastText embeddings have insufficient quality when learned on the smallest forums and / or that domain-finetuned BERT subsumes any positive effect. On the high-resource CQADupStack forums, domain-specific embeddings contribute positively, while generic GloVe does not (rows 1,2). Table 5 shows that omitting either f_G or f_D from the ensemble does not lead to a significant drop in MAP, but omitting both does.

USE has the biggest positive effect on MV-DASE (Table 6, rows 3,11), also evidenced by the fact that omitting it is significantly more harmful than omitting any other single view (Table 5). Recall from Section 3.2 that USE is trained on supervised transfer tasks, while the remaining encoders are fully unsupervised.

GCCA ablation. The naive concatenation or averaging of views is significantly less effective than view correlation by GCCA (Table 6, rows 7,8,15,16, and Table 5). This underlines that multi-view learning is not just about *which* views are combined, but also about *how*. Intuitively, GCCA discovers which features from the different encoders "mean the same thing" in the domain. By contrast, concatenation treats views as orthogonal, while averaging mixes them in an unstructured way.

6 Evaluation on SemEval-2017 3B

In this section, we evaluate MV-DASE on SemEval-2017 3B, a DQD shared task based on the QatarLiving CQA forum. The benchmark provides manually labeled question pairs for training as well as additional unlabeled in-domain data. Since MV-DASE is unsupervised, we discard all training labels and concatenate training and unlabeled data into a text corpus ($\approx$ 1.5M tokens). This corpus is used for FastText training, BERT domain-finetuning, SIF weight estimation and GCCA, as described in Section 3.

The test set contains 88 queries q with ten candidates $c_1 \dots c_{10}$ each. We preprocess all data with the CQADupStack package and concatenate question subjects and bodies, before encoding them. We rank candidates by $\cos(f(q), f(c))$ and evaluate the result with the official shared task scorer.[16] In keeping with the original leaderboard, we report MAP and MRR (Mean Reciprocal Rank). We compare against previous literature as well as all individual views, view concatenation and averaging. See Table 7 for results. Like we observed on the Stack Exchange data, MV-DASE outperforms its individual views, their concatenation and average. It beats the previous State of the Art (a supervised system) by a margin of 2.5% MAP.

7 Evaluation on unsupervised STS

While this paper focuses on Duplicate Question Detection, MV-DASE is also applicable to other unsupervised sentence-pair tasks. As proof of concept, we test it on the unsupervised STS Benchmark (Cer et al., 2017). Here, the task is to predict similarity scores $y \in \mathbb{R}$ for sentence pairs (s_1, s_2).

[16] alt.qcri.org/semeval2017/task3/data/uploads/semeval2017_task3_submissions_and_scores.zip

		MAP	MRR
1	f_G (GloVe)	43.13	47.39
2	f_D (FastText)	43.38	47.67
3	f_U (USE)	48.22	52.73
4	f_B (BERT)	43.51	48.52
5	MV-DASE	51.56	56.48
6	concat	44.66	49.84
7	avg	44.95	49.76
8	Filice et al. (2017)*	49.00	52.41
9	Charlet and Damnati (2017)*	47.87	50.97
10	Goyal (2017)*	47.20	53.22
11	Zhang and Wu (2018)	48.53	52.75
12	Yang et al. (2018)	48.97	-
13	Gonzalez et al. (2018)	48.56	52.41
14	IR baseline*	41.85	46.42
15	Random baseline*	29.81	33.02

Table 7: MAP and MRR (percentages) on SemEval-2017 3B test set. *Shared task top teams (best run out of three) and baselines as reported in Nakov et al. (2017), Table 6. Gray: best in column.

We treat the benchmark training set as an unlabeled corpus, i.e., we discard all labels and destroy the original sentence pairings by shuffling. The resulting corpus is used for BERT domain-finetuning, SIF weight estimation and GCCA. At test time, we measure Pearson's r between $\cos(f(s_1), f(s_2))$ and y, where f is an encoder (e.g., MV-DASE) and y is the ground truth similarity of test set pair (s_1, s_2).

In this experiment, the ensemble contains USE (f_U), domain-finetuned BERT (f_B) and f_G. For f_G, we either use SIF-weighted averaged GloVe vectors (Section 3.2), or unweighted averaged ParaNMT[17] word and trigram vectors (Wieting and Gimpel, 2018), which are the current State of the Art on the unsupervised STS Benchmark test set (Ethayarajh, 2018). The unlabeled training set is very small (64K tokens); hence, we do not include f_D in the ensemble, and we finetune the BERT language model for 10K rather than 100K steps to avoid overfitting. Like on the DQD tasks, MV-DASE beats its individual views as well as naive view concatenation and averaging (see Table 8). After adding ParaNMT to the ensemble, we achieve competitive results.

8 Future Work

Non-Linear GCCA. In Section 3.1, we assumed that relationships between representations are linear. This is probably reasonable for word embeddings (most cross-lingual word embeddings are linear projections, e.g. Artetxe et al. (2018)), but it is unclear whether it holds for sentence embeddings. Potential avenues for non-linear GCCA include Kernel GCCA (Tenenhaus et al., 2015) and Deep GCCA (Benton et al., 2017).

More views. A major advantage of MV-DASE is that it is agnostic to the number and specifics of its views. We plan to investigate whether additional or different views (e.g., encoders learned on related domains) can increase performance.

[17] github.com/jwieting/para-nmt-50m

		f_G = GloVe	f_G = ParaNMT
1	f_G	.731 / .647	.817 / .799
2	f_U (USE)	.793 / .762	.793 / .762
3	f_B (BERT)	.779 / .718	.779 / .718
4	MV-DASE	.825 / .771	.842 / .804
5	concat	.791 / .730	.826 / .772
6	avg	.790 / .729	.823 / .771

Table 8: Pearson's r (dev / test) on the unsupervised STS Benchmark, using different embeddings for f_G. Gray: best in column. Underlined: current unsupervised SoTA on test set (Wieting and Gimpel, 2018).

9 Conclusion

We have presented a multi-view approach to unsupervised Duplicate Question Detection in low-resource, domain-specific Community Question Answering forums. MV-DASE is a multi-view sentence embedding framework based on Generalized Canonical Correlation Analysis. It combines domain-specific and generic weighted averaged word embeddings with domain-finetuned BERT and the Universal Sentence Encoder, using unlabeled in-domain data only.

Experiments on the CQADupStack corpus and additional low-resource forums show significant improvements over BM25 and all single-view baselines. MV-DASE sets a new State of the Art on a recent DQD shared task (SemEval-2017 3B), with a 2.5% MAP improvement over the best supervised system. Finally, an experiment on the STS Benchmark suggests that MV-DASE has potential on other unsupervised sentence-pair tasks.

Acknowledgements

We thank Bernt Andrassy and Pankaj Gupta at Siemens MIC-DE, as well as our anonymous reviewers, for their helpful comments. This research was funded by Siemens AG.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, Toulon, France.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *ACL*, pages 789–798, Melbourne, Australia.

Francis R Bach and Michael I Jordan. 2002. Kernel independent component analysis. *JMLR*, 3:1–48.

Timothy Baldwin, Huizhi Liang, Bahar Salehi, Doris Hoogeveen, Yitong Li, and Long Duong. 2016. UniMelb at SemEval-2016 Task 3: Identifying similar questions by combining a CNN with string similarity measures. In *International Workshop on Semantic Evaluation*, pages 851–856, San Diego, USA.

Adrian Benton, Huda Khayrallah, Biman Gujral, Dee Ann Reisinger, Sheng Zhang, and Raman Arora. 2017. Deep generalized canonical correlation analysis. *arXiv preprint arXiv:1702.02519*.

Dasha Bogdanova, Cicero dos Santos, Luciano Barbosa, and Bianca Zadrozny. 2015. Detecting semantically equivalent questions in online user forums. In *CoNLL*, pages 123–131, Beijing, China.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Danushka Bollegala and Cong Bao. 2018. Learning word meta-embeddings by autoencoding. In *COLING*, pages 1650–1661, Santa Fe, USA.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *ACL*, pages 730–740, Beijing, China.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, Lisbon, Portugal.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *International Workshop on Semantic Evaluation*, pages 1–14, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder for English. In *EMNLP*, pages 169–174, Brussels, Belgium.

Delphine Charlet and Geraldine Damnati. 2017. SimBow at SemEval-2017 Task 3: Soft-cosine semantic similarity between questions for community question answering. In *International Workshop on Semantic Evaluation*, pages 315–319, Vancouver, Canada.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *NAACL-HLT*, pages 1226–1240, New Orleans, USA.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *NAACL-HLT*, pages 194–198, New Orleans, USA.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, USA.

Cicero Dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *ACL*, pages 694–699, Beijing, China.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Workshop on Representation Learning for NLP*, pages 91–100, Melbourne, Australia.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471, Gothenburg, Sweden.

Simone Filice, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. 2017. KeLP at SemEval-2017 Task 3: Learning pairwise patterns in community question answering. In *International Workshop on Semantic Evaluation*, pages 326–333, Vancouver, Canada.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*, 17:2096–2030.

Ana Gonzalez, Isabelle Augenstein, and Anders Søgaard. 2018. A strong baseline for question relevancy ranking. In *EMNLP*, pages 4810–4815, Brussels, Belgium.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media.

Naman Goyal. 2017. LearningToQuestion at semeval 2017 Task 3: Ranking similar questions by learning to rank using rich features. In *International Workshop on Semantic Evaluation*, pages 326–333, Vancouver, Canada.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

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

Doris Hoogeveen, Andrew Bennett, Yitong Li, Karin M Verspoor, and Timothy Baldwin. 2018. Detecting misflagged duplicate questions in community question-answering archives. In *ICWSM*, Stanford, USA.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In *Australasian Document Computing Symposium*, Parramatta, Australia.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2016. CQADupStack: Gold or silver. In *Workshop on Web Question Answering Beyond Factoids*, volume 16, Pisa, Italy.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339, Melbourne, Australia.

Jon R Kettenring. 1971. Canonical analysis of several sets of variables. *Biometrika*, 58(3):433–451.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *EMNLP*, pages 1466–1477, Brussels, Belgium.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NeurIPS*, pages 3294–3302, Montreal, Canada.

Jey Hand Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, Beijing, China.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT*, pages 1073–1094, Minneapolis, USA.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *ICLR*, Vancouver, Canada.

Preslav Nakov, Doris Hoogeveen, Lluıs Marquez, Allessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community question answering. In *International Workshop on Semantic Evaluation*, pages 27–48, Vancouver, Canada.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237, New Orleans, USA.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *NAACL-HLT*, pages 556–566, Denver, USA.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Text REtrieval Conference (TREC)*, pages 109–126.

João António Rodrigues, Chakaveh Saedi, Vladislav Maraev, Joao Silva, and António Branco. 2017. Ways of asking and replying in duplicate question detection. In *Joint Conference on Lexical and Computational Semantics*, pages 262–270, Vancouver, Canada.

Prathusha K Sarma, Yingyu Liang, and William A Sethares. 2018. Domain adapted word embeddings for improved sentiment classification. In *Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 51–59, Melbourne, Australia.

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *EMNLP*, pages 1056–1063, Brussels, Belgium.

Shuai Tang and Virginia R de Sa. 2019. Improving sentence representations with multi-view frameworks. In *Interpretability and Robustness for Audio, Speech and Language Workshop*, Montreal, Canada.

Arthur Tenenhaus, Cathy Philippe, and Vincent Frouin. 2015. Kernel generalized canonical correlation analysis. *Computational Statistics & Data Analysis*, 90:114–131.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*, pages 451–462, Melbourne, Australia.

Wei Yang, Wei Lu, and Vincent W Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *EMNLP*, pages 2898–2904, Copenhagen, Denmark.

Ziyi Yang, Chenguang Zhu, and Weizhu Chen. 2018. Zero-training sentence embedding via orthogonal basis. *arXiv preprint arXiv:1810.00438*.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *ACL*, pages 1351–1360, Berlin, Germany.

Mingua Zhang and Yunfang Wu. 2018. An unsupervised model with attention autoencoders for question retrieval. In *AAAI*, pages 4978–4986, New Orleans, USA.

Wei Emma Zhang, Quan Z Sheng, Jey Han Lau, and Ermyas Abebe. 2017. Detecting duplicate posts in programming QA communities via latent semantics and association rules. In *WWW*, pages 1221–1229.

Chapter 5

Corresponds to the following publication:

> Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. Sentence meta-embeddings for unsupervised semantic textual similarity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7027–7034, Online

Declaration of Co-Authorship: I conceived of the original research, implemented the model and performed the evaluation experiments. I wrote the initial draft of the paper. My supervisor, Hinrich Schütze, contributed through continuous discussions in our weekly meetings, and by reviewing the paper before submission. Ulli Waltinger contributed by reviewing an earlier draft of the paper.

[1] `https://www.aclweb.org/anthology/faq/copyright/` [accessed 05/03/2021]

Sentence Meta-Embeddings for Unsupervised Semantic Textual Similarity

Nina Poerner[*†] **and Ulli Waltinger**[†] **and Hinrich Schütze**[*]
[*]Center for Information and Language Processing, LMU Munich, Germany
[†]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

Abstract

We address the task of unsupervised Semantic Textual Similarity (STS) by ensembling diverse pre-trained sentence encoders into *sentence meta-embeddings*. We apply, extend and evaluate different meta-embedding methods from the word embedding literature at the sentence level, including dimensionality reduction (Yin and Schütze, 2016), generalized Canonical Correlation Analysis (Rastogi et al., 2015) and cross-view auto-encoders (Bollegala and Bao, 2018). Our sentence meta-embeddings set a new unsupervised State of The Art (SoTA) on the STS Benchmark and on the STS12–STS16 datasets, with gains of between 3.7% and 6.4% Pearson's r over single-source systems.

1 Introduction

Word meta-embeddings have been shown to exceed single-source word embeddings on word-level semantic benchmarks (Yin and Schütze, 2016; Bollegala and Bao, 2018). Presumably, this is because they combine the complementary strengths of their components.

There has been recent interest in pre-trained "universal" sentence encoders, i.e., functions that encode diverse semantic features of sentences into fixed-size vectors (Conneau et al., 2017). Since these sentence encoders differ in terms of their architecture and training data, we hypothesize that their strengths are also complementary and that they can benefit from meta-embeddings.

To test this hypothesis, we adapt different meta-embedding methods from the word embedding literature. These include dimensionality reduction (Yin and Schütze, 2016), cross-view autoencoders (Bollegala and Bao, 2018) and Generalized Canonical Correlation Analysis (GCCA) (Rastogi et al., 2015). The latter method was also used by Poerner and Schütze (2019) for domain-specific Duplicate Question Detection.

Our sentence encoder ensemble includes three models from the recent literature: Sentence-BERT (Reimers and Gurevych, 2019), the Universal Sentence Encoder (Cer et al., 2017) and averaged ParaNMT vectors (Wieting and Gimpel, 2018). Our meta-embeddings outperform every one of their constituent single-source embeddings on STS12–16 (Agirre et al., 2016) and on the STS Benchmark (Cer et al., 2017). Crucially, since our meta-embeddings are agnostic to the contents of their ensemble, future improvements may be possible by adding new encoders.

2 Related work

2.1 Word meta-embeddings

Word embeddings are functions that map word types to vectors. They are typically trained on unlabeled corpora and capture word semantics (e.g., Mikolov et al. (2013); Pennington et al. (2014)).

Word meta-embeddings combine ensembles of word embeddings by various operations: Yin and Schütze (2016) use concatenation, SVD and linear projection, Coates and Bollegala (2018) show that averaging word embeddings has properties similar to concatenation. Rastogi et al. (2015) apply generalized canonical correlation analysis (GCCA) to an ensemble of word vectors. Bollegala and Bao (2018) learn word meta-embeddings using autoencoder architectures. Neill and Bollegala (2018) evaluate different loss functions for autoencoder word meta-embeddings, while Bollegala et al. (2018) explore locally linear mappings.

2.2 Sentence embeddings

Sentence embeddings are methods that produce one vector per sentence. They can be grouped into two categories:

104 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7027–7034
July 5 - 10, 2020.

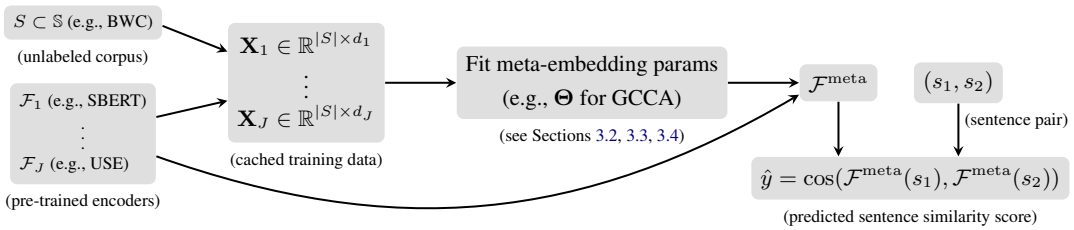

Figure 1: Schematic depiction: Trainable sentence meta-embeddings for unsupervised STS.

(a) Word embedding average sentence encoders take a (potentially weighted) average of pre-trained word embeddings. Despite their inability to understand word order, they are surprisingly effective on sentence similarity tasks (Arora et al., 2017; Wieting and Gimpel, 2018; Ethayarajh, 2018)

(b) Complex contextualized sentence encoders, such as Long Short Term Memory Networks (LSTM) (Hochreiter and Schmidhuber, 1997) or Transformers (Vaswani et al., 2017). Contextualized encoders can be pre-trained as unsupervised language models (Peters et al., 2018; Devlin et al., 2019), but they are usually improved on supervised transfer tasks such as Natural Language Inference (Bowman et al., 2015).

2.3 Sentence meta-embeddings

Sentence meta-embeddings have been explored less frequently than their word-level counterparts. Kiela et al. (2018) create meta-embeddings by training an LSTM sentence encoder on top of a set of dynamically combined word embeddings. Since this approach requires labeled data, it is not applicable to unsupervised STS.

Tang and de Sa (2019) train a Recurrent Neural Network (RNN) and a word embedding average encoder jointly on a large corpus to predict similar representations for neighboring sentences. Their approach trains both encoders from scratch, i.e., it cannot be used to combine existing encoders.

Poerner and Schütze (2019) propose a GCCA-based multi-view sentence encoder that combines domain-specific and generic sentence embeddings for unsupervised Duplicate Question Detection. In this paper, we extend their approach by exploring a wider range of meta-embedding methods and an ensemble that is more suited to STS.

2.4 Semantic Textual Similarity (STS)

Semantic Textual Similarity (STS) is the task of rating the similarity of two natural language sentences on a real-valued scale. Related applications are semantic search, duplicate detection and sentence clustering.

Supervised SoTA systems for STS typically apply cross-sentence attention (Devlin et al., 2019; Raffel et al., 2019). This means that they do not scale well to many real-world tasks. Supervised "siamese" models (Reimers and Gurevych, 2019) on the other hand, while not competitive with cross-sentence attention, can cache sentence embeddings independently of one another. For instance, to calculate the pairwise similarities of N sentences, a cross-sentence attention system must calculate $O(N^2)$ slow sentence pair embeddings, while the siamese model calculates $O(N)$ slow sentence embeddings and $O(N^2)$ fast vector similarities.

Our meta-embeddings are also cacheable (and hence scalable), but they do not require supervision.

3 Sentence meta-embedding methods

Below, we assume access to an ensemble of pre-trained sentence encoders, denoted $\mathcal{F}_1 \dots \mathcal{F}_J$. Every $\mathcal{F}_j$ maps from the (infinite) set of possible sentences $\mathbb{S}$ to a fixed-size d_j-dimensional vector.

Word meta-embeddings are usually learned from a finite vocabulary of word types (Yin and Schütze, 2016). Sentence embeddings lack such a "vocabulary", as they can encode any member of $\mathbb{S}$. Therefore, we train on a sample $S \subset \mathbb{S}$, i.e., on a corpus of unlabeled sentences.

3.1 Naive meta-embeddings

We create naive sentence meta-embeddings by concatenating (Yin and Schütze, 2016) or averaging[1] (Coates and Bollegala, 2018) sentence embeddings.

$$\mathcal{F}^{\text{conc}}(s') = \begin{bmatrix} \hat{\mathcal{F}}_1(s') \\ \dots \\ \hat{\mathcal{F}}_J(s') \end{bmatrix}$$

[1] If embeddings have different dimensionalities, we pad the shorter ones with zeros.

$$\mathcal{F}^{\text{avg}}(s') = \sum_j \frac{\hat{\mathcal{F}}_j(s')}{J}$$

Note that we length-normalize all embeddings to ensure equal weighting:

$$\hat{\mathcal{F}}_j(s) = \frac{\mathcal{F}_j(s)}{||\mathcal{F}_j(s)||_2}$$

3.2 SVD

Yin and Schütze (2016) use Singular Value Decomposition (SVD) to compactify concatenated word embeddings. The method is straightforward to extend to sentence meta-embeddings. Let $\mathbf{X}^{\text{conc}} \in \mathbb{R}^{|S| \times \sum_j d_j}$ with

$$\mathbf{x}_n^{\text{conc}} = \mathcal{F}^{\text{conc}}(s_n) - \mathbb{E}_{s \in S}[\mathcal{F}^{\text{conc}}(s)]$$

Let $\mathbf{U}\mathbf{S}\mathbf{V}^T \approx \mathbf{X}^{\text{conc}}$ be the d-truncated SVD. The SVD meta-embedding of a new sentence s' is:

$$\mathcal{F}^{\text{svd}}(s') = \mathbf{V}^T(\mathcal{F}^{\text{conc}}(s') - \mathbb{E}_{s \in S}[\mathcal{F}^{\text{conc}}(s)])$$

3.3 GCCA

Given random vectors $\mathbf{x}_1, \mathbf{x}_2$, Canonical Correlation Analysis (CCA) finds linear projections such that $\boldsymbol{\theta}_1^T \mathbf{x}_1$ and $\boldsymbol{\theta}_2^T \mathbf{x}_2$ are maximally correlated. Generalized CCA (GCCA) extends CCA to three or more random vectors. Bach and Jordan (2002) show that a variant of GCCA reduces to a generalized eigenvalue problem on block matrices:

$$\rho \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\dots} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_{J,J} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_1 \\ \dots \\ \boldsymbol{\theta}_J \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{\dots} & \boldsymbol{\Sigma}_{1,J} \\ \boldsymbol{\Sigma}_{\dots} & \mathbf{0} & \boldsymbol{\Sigma}_{\dots} \\ \boldsymbol{\Sigma}_{J,1} & \boldsymbol{\Sigma}_{\dots} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_1 \\ \dots \\ \boldsymbol{\theta}_J \end{bmatrix}$$

where

$$\boldsymbol{\Sigma}_{j,j'} = \mathbb{E}_{s \in S}[(\mathcal{F}_j(s) - \boldsymbol{\mu}_j)(\mathcal{F}_{j'}(s) - \boldsymbol{\mu}_{j'})^T]$$
$$\boldsymbol{\mu}_j = \mathbb{E}_{s \in S}[\mathcal{F}_j(s)]$$

For stability, we add $\frac{\tau}{d_j} \sum_{n=1}^{d_j} \text{diag}(\boldsymbol{\Sigma}_{j,j})_n$ to $\text{diag}(\boldsymbol{\Sigma}_{j,j})$, where τ is a hyperparameter. We stack the eigenvectors of the top-d eigenvalues into matrices $\boldsymbol{\Theta}_j \in \mathbb{R}^{d \times d_j}$ and define the GCCA meta-embedding of sentence s' as:

$$\mathcal{F}^{\text{gcca}}(s') = \sum_{j=1}^{J} \boldsymbol{\Theta}_j(\mathcal{F}_j(s') - \boldsymbol{\mu}_j)$$

$\mathcal{F}^{\text{gcca}}$ corresponds to MV-DASE in Poerner and Schütze (2019).

		loss function			
		MSE	MAE	KLD	(1-COS)^2
number hidden layers	0	83.0/84.2	84.2/85.1	83.0/84.2	82.4/83.5
	1	82.7/83.9	83.8/84.6	**85.1/85.5**	83.3/83.4
	2	82.5/82.8	81.3/82.1	83.3/83.4	82.3/82.3

$\tau = 10^{-2}$	$\tau = 10^{-1}$	$\tau = 10^{0}$	$\tau = 10^{1}$	$\tau = 10^{2}$
84.2/84.1	84.8/84.7	85.5/85.7	**85.5/86.1**	84.9/85.9

Table 1: Hyperparameter search on STS Benchmark development set for AE (top) and GCCA (bottom). Pearson's $r \times 100$ / Spearman's $\rho \times 100$.

3.4 Autoencoders (AEs)

Autoencoder meta-embeddings are trained by gradient descent to minimize some cross-embedding reconstruction loss. For example, Bollegala and Bao (2018) train feed-forward networks (FFN) to encode two sets of word embeddings into a shared space, and then reconstruct them such that mean squared error with the original embeddings is minimized. Neill and Bollegala (2018) evaluate different reconstruction loss functions: Mean Squared Error (MSE), Mean Absolute Error (MAE), KL-Divergence (KLD) or squared cosine distance (1-COS)^2.

We extend their approach to sentence encoders as follows: Every sentence encoder $\mathcal{F}_j$ has a trainable encoder $\mathcal{E}_j : \mathbb{R}^{d_j} \rightarrow \mathbb{R}^d$ and a trainable decoder $\mathcal{D}_j : \mathbb{R}^d \rightarrow \mathbb{R}^{d_j}$, where d is a hyperparameter. Our training objective is to reconstruct every embedding $\mathbf{x}_{j'}$ from every $\mathcal{E}_j(\mathbf{x}_j)$. This results in J^2 loss terms, which are jointly optimized:

$$L(\mathbf{x}_1 \dots \mathbf{x}_J) = \sum_j \sum_{j'} l(\mathbf{x}_{j'}, \mathcal{D}_{j'}(\mathcal{E}_j(\mathbf{x}_j)))$$

where l is one of the reconstruction loss functions listed above. The autoencoder meta-embedding of a new sentence s' is:

$$\mathcal{F}^{\text{ae}}(s') = \sum_j \mathcal{E}_j(\mathcal{F}_j(s'))$$

4 Experiments

4.1 Data

We train on all sentences of length < 60 from the first file (*news.en-00001-of-00100*) of the tokenized, lowercased Billion Word Corpus (BWC) (Chelba et al., 2014) ($\sim$302K sentences). We evaluate on STS12 – STS16 (Agirre et al., 2016) and the unsupervised STS Benchmark test set (Cer et al.,

		dimensionality	STS12	STS13	STS14	STS15	STS16	STS-B
single:ParaNMT		$d = 600$	67.5/66.3	62.7/62.8	77.3/74.9	80.3/80.8	78.3/79.1	79.8/78.9
single:USE		$d = 512$	62.6/63.8	57.3/57.8	69.5/66.0	74.8/77.1	73.7/76.4	76.2/74.6
single:SBERT		$d = 1024$	66.9/66.8	63.2/64.8	74.2/74.3	77.3/78.3	72.8/75.7	76.2/79.2
single:ParaNMT – up-projection*		$d = 1024$	67.3/66.2	62.1/62.4	77.1/74.7	79.7/80.2	77.9/78.7	79.5/78.6
single:USE – up-projection*		$d = 1024$	62.4/63.7	57.0/57.5	69.4/65.9	74.7/77.1	73.6/76.3	76.0/74.5
meta:conc		$d = 2136$	72.7/71.3	68.4/68.6	81.0/79.0	84.1/**85.5**	**82.0/83.8**	82.8/83.4
meta:avg		$d = 1024$	72.5/71.2	68.1/68.3	80.8/78.8	83.7/85.1	81.9/83.6	82.5/83.2
meta:svd		$d = 1024$	71.9/70.8	68.3/68.3	80.6/78.6	83.8/85.1	81.6/83.6	83.4/83.8
meta:gcca	(hyperparams on dev set)	$d = 1024$	**72.8/71.6**	**69.6/69.4**	**81.7/79.5**	**84.2/85.5**	81.3/83.3	**83.9/84.4**
meta:ae	(hyperparams on dev set)	$d = 1024$	71.5/70.6	68.5/68.4	80.1/78.5	82.5/83.1	80.4/81.9	82.1/83.3
Ethayarajh (2018)		(unsupervised)	68.3/-	66.1/-	78.4/-	79.0/-	-/-	79.5/-
Wieting and Gimpel (2018)		(unsupervised)	68.0/-	62.8/-	77.5/-	80.3/-	78.3/-	79.9/-
Tang and de Sa (2019)		(unsupervised meta)	64.0/-	61.7/-	73.7/-	77.2/-	76.7/-	-
Hassan et al. (2019)†		(unsupervised meta)	67.7/-	64.6/-	75.6/-	80.3/-	79.3/-	77.7/-
Poerner and Schütze (2019)		(unsupervised meta)	-/-	-/-	-/-	-/-	-/-	80.4/-
Reimers and Gurevych (2019)		(sup. siamese SoTA)	-/-	-/-	-/-	-/-	-/-	-/86.2
Raffel et al. (2019)		(supervised SoTA)	-/-	-/-	-/-	-/-	-/-	93.1/92.8

Table 2: Results on STS12–16 and STS Benchmark test set. STS12–16: mean Pearson's $r \times 100$ / Spearman's $\rho \times 100$. STS Benchmark: overall Pearson's $r \times 100$ / Spearman's $\rho \times 100$. Evaluated by SentEval (Conneau and Kiela, 2018). **Boldface:** best in column (except supervised). Underlined: best single-source method. *Results for up-projections are averaged over 10 random seeds. †Unweighted average computed from Hassan et al. (2019, Table 8). There is no supervised SoTA on STS12–16, as they are unsupervised benchmarks.

2017).[2] These datasets consist of triples (s_1, s_2, y), where s_1, s_2 are sentences and y is their ground truth semantic similarity. The task is to predict similarity scores $\hat{y}$ that correlate well with y. We predict $\hat{y} = \cos(\mathcal{F}(s_1), \mathcal{F}(s_2))$.

4.2 Metrics

Previous work on STS differs with respect to (a) the correlation metric and (b) how to aggregate the subtestsets of STS12–16. To maximize comparability, we report both Pearson's r and Spearman's ρ. On STS12–16, we aggregate by a non-weighted average, which diverges from the original shared tasks (Agirre et al., 2016) but ensures comparability with more recent baselines (Wieting and Gimpel, 2018; Ethayarajh, 2018). Results for individual STS12–16 sub-testsets can be found in the Appendix.

4.3 Ensemble

We select our ensemble according to the following criteria: Every encoder should have near-SoTA performance on the unsupervised STS benchmark, and the encoders should not be too similar with regards to their training regime. For instance, we do not use Ethayarajh (2018), which is a near-SoTA unsupervised method that uses the same word vectors as ParaNMT (see below).

We choose the Universal Sentence Encoder (USE)[3] (Cer et al., 2018), which is a Transformer trained on skip-thought, conversation response prediction and Natural Language Inference (NLI), Sentence-BERT (SBERT)[4] (Reimers and Gurevych, 2019), which is a pre-trained BERT transformer finetuned on NLI, and ParaNMT[5] (Wieting and Gimpel, 2018), which averages word and 3-gram vectors trained on backtranslated similar sentence pairs. To our knowledge, ParaNMT is the current single-source SoTA on the unsupervised STS Benchmark.

4.4 Hyperparameters

We set $d = 1024$ in all experiments, which corresponds to the size of the biggest single-source embedding (SBERT). The value of τ (GCCA), as well as the autoencoder depth and loss function are tuned on the STS Benchmark development set (see

[2] We use SentEval for evaluation (Conneau and Kiela, 2018). Since original SentEval does not support the unsupervised STS Benchmark, we use a non-standard repository (https://github.com/sidak/SentEval). We manually add the missing STS13-SMT subtask.

[3] https://tfhub.dev/google/universal-sentence-encoder/2

[4] https://github.com/UKPLab/sentence-transformers. We use the *large-nli-mean-tokens* model, which was **not** finetuned on STS.

[5] https://github.com/jwieting/para-nmt-50m

	full ensemble	without ParaNMT	without USE	without SBERT
meta:svd	**85.0/85.4**	79.6/81.3	79.7/81.4	83.7/83.5
meta:gcca	**85.5/86.1**	84.9/84.8	83.8/83.8	85.4/85.4
meta:ae	**85.1/85.5**	76.5/80.3	82.5/83.5	28.7/41.0

Table 3: Ablation study: Pearson's $r \times 100$ / Spearman's $\rho \times 100$ on STS Benchmark development set when one encoder is left out.

Table 1). We train the autoencoder for a fixed number of 500 epochs with a batch size of 10,000. We use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and learning rate 0.001.

4.5 Baselines

Our main baselines are our single-source embeddings. Wieting and Kiela (2019) warn that high-dimensional sentence representations can have an advantage over low-dimensional ones, i.e., our meta-embeddings might be better than lower-dimensional single-source embeddings due to size alone. To exclude this possibility, we also up-project smaller embeddings by a random $d \times d_j$ matrix sampled from:

$$\mathcal{U}(-\frac{1}{\sqrt{d_j}}, \frac{1}{\sqrt{d_j}})$$

Since the up-projected sentence embeddings perform slightly worse than their originals (see Table 2, rows 4–5), we are confident that performance gains by our meta-embeddings are due to content rather than size.

4.6 Results

Table 2 shows that even the worst of our meta-embeddings consistently outperform their single-source components. This underlines the overall usefulness of ensembling sentence encoders, irrespective of the method used.

GCCA outperforms the other meta-embeddings on five out of six datasets. We set a new unsupervised SoTA on the unsupervised STS Benchmark test set, reducing the gap with the supervised siamese SoTA of Reimers and Gurevych (2019) from 7% to 2% Spearman's ρ.

Interestingly, the naive meta-embedding methods (concatenation and averaging) are competitive with SVD and the autoencoder, despite not needing any unsupervised training. In the case of concatenation, this comes at the cost of increased dimensionality, which may be problematic for downstream applications. The naive averaging method by Coates and Bollegala (2018) however does not have this problem, while performing only marginally worse than concatenation.

4.7 Ablation

Table 3 shows that all single-source embeddings contribute positively to the meta-embeddings, which supports their hypothesized complementarity. This result also suggests that further improvements may be possible by extending the ensemble.

4.8 Computational cost

4.8.1 Training

All of our meta-embeddings are fast to train, either because they have closed-form solutions (GCCA and SVD) or because they are lightweight feed-forward nets (autoencoder). The underlying sentence encoders are more complex and slow, but since we do not update them, we can apply them to the unlabeled training data once and then reuse the results as needed.

4.8.2 Inference

As noted in Section 2.4, cross-sentence attention systems do not scale well to many real-world STS-type tasks, as they do not allow individual sentence embeddings to be cached. Like Reimers and Gurevych (2019), our meta-embeddings do not have this problem. This should make them more suitable for tasks like sentence clustering or real-time semantic search.

5 Conclusion

Inspired by the success of word meta-embeddings, we have shown how to apply different meta-embedding techniques to ensembles of sentence encoders. All sentence meta-embeddings consistently outperform their individual single-source components on the STS Benchmark and the STS12–16 datasets, with a new unsupervised SoTA set by our GCCA meta-embeddings. Because sentence meta-embeddings are agnostic to the size and specifics of their ensemble, it should be possible to add new encoders to the ensemble, potentially improving performance further.

Acknowledgments. This work was supported by Siemens AG and by the European Research Council (# 740516).

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *International Workshop on Semantic Evaluation*, pages 497–511, San Diego, USA.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, Toulon, France.

Francis R Bach and Michael I Jordan. 2002. Kernel independent component analysis. *JMLR*, 3:1–48.

Danushka Bollegala and Cong Bao. 2018. Learning word meta-embeddings by autoencoding. In *COLING*, pages 1650–1661, Santa Fe, USA.

Danushka Bollegala, Kohei Hayashi, and Ken-ichi Kawarabayashi. 2018. Think globally, embed locally – locally linear meta-embedding of words. In *ICJAI*, pages 3970–3976, Stockholm, Sweden.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, Lisbon, Portugal.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *International Workshop on Semantic Evaluation*, pages 1–14, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder for English. In *EMNLP*, pages 169–174, Brussels, Belgium.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*, pages 2635–2639, Singapore.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *NAACL-HLT*, pages 194–198, New Orleans, USA.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *LREC*, pages 1699–1704, Miyazaki, Japan.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, New Orleans, USA.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Workshop on Representation Learning for NLP*, pages 91–100, Melbourne, Australia.

Basma Hassan, Samir E Abdelrahman, Reem Bahgat, and Ibrahim Farag. 2019. UESTS: An unsupervised ensemble semantic textual similarity method. *IEEE Access*, 7:85462–85482.

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

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *EMNLP*, pages 1466–1477, Brussels, Belgium.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, Lake Tahoe, USA.

James O' Neill and Danushka Bollegala. 2018. Angular-based word meta-embedding learning. *arXiv preprint arXiv:1808.04334*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237, New Orleans, USA.

Nina Poerner and Hinrich Schütze. 2019. Multi-view domain adapted sentence embeddings for low-resource unsupervised duplicate question detection. In *EMNLP-IJCNLP*, Hong Kong, China.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *NAACL-HLT*, pages 556–566, Denver, USA.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP-IJCNLP*, Hong Kong, China.

Shuai Tang and Virginia R de Sa. 2019. Improving sentence representations with multi-view frameworks. In *Interpretability and Robustness for Audio, Speech and Language Workshop*, Montreal, Canada.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*, pages 451–462, Melbourne, Australia.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *ICLR*, New Orleans, USA.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *ACL*, pages 1351–1360, Berlin, Germany.

	single-source embeddings			meta-embeddings				
method:	ParaNMT	SBERT	USE	conc	avg	svd	gcca	ae
dimensionality:	$d = 600$	$d = 1024$	$d = 512$	$d = 2136$	$d = 1024$	$d = 1024$	$d = 1024$	$d = 1024$
STS12								
MSRpar	55.25/55.15	58.11/60.42	34.05/39.24	60.13/60.53	58.90/59.71	59.56/60.24	**62.79/63.90**	61.64/63.57
MSRvid	88.53/88.48	87.93/89.73	89.46/90.75	**91.51**/92.16	91.29/91.92	91.28/91.98	91.20/**92.29**	90.69/91.69
SMTeuroparl	53.15/59.31	59.63/62.40	49.00/62.08	58.99/64.02	**60.16/64.73**	57.03/62.17	56.40/61.23	55.13/60.14
OnWN	73.42/69.82	68.08/68.51	71.66/65.81	77.89/73.05	77.53/73.00	77.80/73.12	**77.90/73.50**	75.35/73.03
SMTnews	67.03/58.53	60.75/53.11	68.66/61.29	74.85/66.53	74.54/66.88	73.73/66.48	**75.75/67.31**	74.91/64.76
STS13								
FNWN	53.01/54.44	57.06/57.22	48.07/49.34	**64.11/64.91**	63.46/64.26	63.28/63.49	62.74/63.54	63.99/64.61
OnWN	75.62/75.80	77.54/80.00	66.64/68.10	80.84/81.13	80.46/80.81	79.89/80.53	**84.04/83.65**	80.17/81.50
SMT	42.54/41.13	44.54/44.80	43.85/41.80	47.46/44.89	47.87/45.04	48.59/45.58	**49.20/46.01**	48.92/45.40
headlines	79.52/79.83	73.67/77.17	70.70/71.82	81.13/83.48	80.64/82.96	81.49/83.54	**82.58/84.37**	80.78/82.13
STS14								
OnWN	82.22/83.20	81.51/82.99	74.61/76.01	85.08/ 85.83	85.12/85.84	84.23/85.17	**87.34/87.27**	84.24/85.09
deft-forum	60.01/59.49	57.66/60.45	50.12/49.43	67.57/66.84	67.09/66.19	66.84/66.20	**68.40/67.26**	67.22/66.82
deft-news	77.46/72.75	72.62/76.80	68.35/63.35	**81.72**/79.04	81.60/78.98	80.36/78.31	81.09/**79.20**	79.59/78.83
headlines	78.85/76.98	73.72/75.41	65.88/62.34	79.64/79.93	79.39/79.86	79.85/79.59	**81.68/81.50**	80.13/79.77
images	86.14/83.36	84.57/79.42	85.54/80.55	**89.52/85.68**	89.35/85.51	89.29/85.37	88.83/84.83	87.64/83.42
tweet-news	79.39/73.43	75.12/70.80	72.48/64.24	82.50/76.50	82.12/76.13	**83.14**/77.17	83.09/77.04	81.61/**77.23**
STS15								
answers-forums	73.54/74.50	64.04/62.78	72.70/75.02	**79.33/79.91**	78.47/79.12	79.15/79.69	78.39/78.59	72.65/72.21
answers-stud.	77.06/77.87	79.12/80.14	60.99/63.32	81.01/82.10	80.15/81.45	81.02/82.14	80.86/82.18	**83.03/83.56**
belief	80.28/80.25	77.46/77.46	78.68/82.14	86.14/**87.58**	85.55/87.01	85.05/86.02	**86.38/87.58**	82.49/83.07
headlines	81.92/82.28	78.91/81.88	73.26/74.77	83.20/86.03	83.33/86.25	83.48/86.02	**84.87/86.72**	84.16/85.53
images	88.60/88.87	86.76/89.02	88.39/90.34	**90.92/91.95**	90.86/91.92	90.46/91.59	90.34/91.85	90.26/91.35
STS16								
answer-answer	69.71/68.96	63.41/66.63	72.52/72.72	**79.65**/78.89	78.93/77.82	79.37/**79.21**	78.70/78.50	76.83/77.17
headlines	80.47/81.90	75.23/79.33	69.70/75.11	80.97/**84.95**	80.60/84.53	81.36/85.14	**81.41**/84.85	80.40/83.17
plagiarism	84.49/85.62	80.78/82.04	74.93/77.42	85.86/87.17	85.88/87.25	85.54/87.36	**85.92/87.76**	85.01/86.14
postediting	84.53/86.34	81.32/85.87	82.81/86.49	**88.18/90.76**	87.98/90.51	87.55/90.21	87.01/90.24	86.71/89.28
question-quest.	72.37/72.73	63.38/64.72	68.54/70.25	75.49/77.42	**76.05/77.76**	74.08/75.93	73.44/74.98	73.25/73.60

Table 4: Pearson's r / Spearman's $\rho \times 100$ on individual sub-testsets of STS12–STS16. **Boldface**: best method in row.