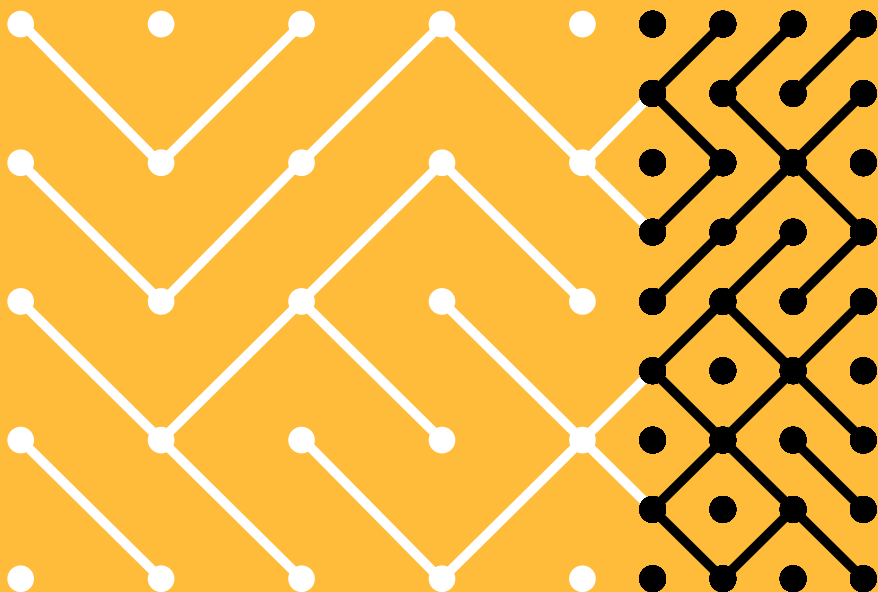


Transfer Learning for Natural Language Processing



Cloudera Fast Forward Labs

Transfer Learning for Natural Language Generation

CLOUDERA **FF**

Copyright © 2019 by Cloudera Fast Forward Labs

<http://fastforwardlabs.com>

New York, NY

To the future—

Contents

1 Introduction	7
2 Background	11
Algorithmic Understanding of Language Is Difficult	11
We Need Models That Understand Language	12
Modeling: Approaches to Representing Language	16
Practical Challenges to Building Sequence Models	22
What Is Transfer Learning and Why Does It Help?	23
3 Technical	31
The Structure of NLP Models	31
Transferring Token Embeddings	34
Transferring Contextual Token Embeddings	36
How Do You Actually Use Transfer Learning?	43
Model Variants	49
4 Prototype	55
Sentiment	55
Dataset	56
Models	57
Interpretability	65
Successes and Failures	71
Product Design: Textflix	77

5 Landscape	83
<i>Use Cases</i>	83
<i>Vendors</i>	88
<i>Tools</i>	89
6 Ethics	95
<i>Model Poisoning</i>	95
<i>Privacy</i>	96
<i>Spoofing</i>	97
<i>Regulation</i>	98
7 Future	101
<i>Understanding and Bias</i>	101
<i>Beyond Syntax</i>	103
<i>Multilingual Models</i>	104
<i>Sci-fi Story: An Introduction to Empathy</i>	107
8 Conclusion	111

CHAPTER 1

Introduction

Natural language, written or spoken, is the way in which humans communicate with one another, organize our thoughts, and describe the world around us. It is the most direct and unfiltered representation of our interactions with the world. It is, in essence, the data of humans.

We're already capable of building machine learning systems that learn from data, but natural language presents

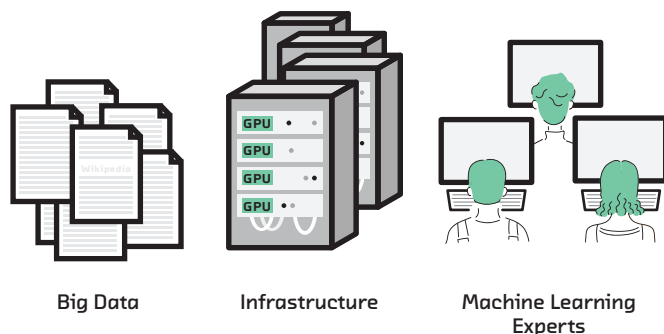


FIGURE 1.1 Building intelligent language processing systems with current methods requires large labeled datasets, costly infrastructure, and scarce machine learning experts.

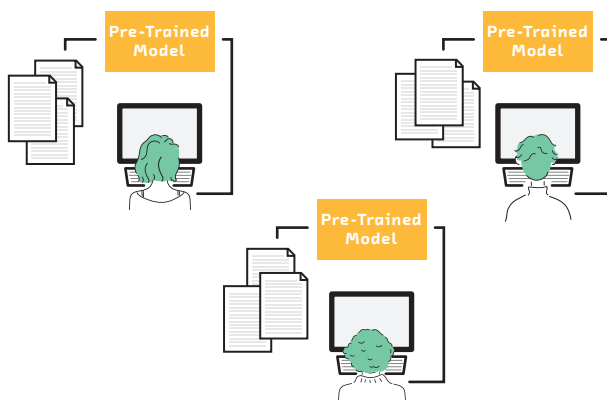


FIGURE 1.2 *Transfer learning allows data scientists to reuse state-of-the-art models, without incurring the steep costs, for a variety of language applications.*

particular challenges. Most data, like GPS signals, temperature measurements, or images, is numeric. But machines do not understand language as they do numbers. Language is an ordered but idiosyncratic collection of symbols (characters, words, and punctuation). It is unstructured, and rarely meaningful in isolation. Meanings are often unstated, or only make sense in a larger context. It is thus challenging to build machine learning systems that understand the meaning of language.

The application of deep learning to natural language processing (NLP) enables us to build *sequence models* that process language as it occurs naturally—as a sequence of symbols, preserving order and structure and incorporating context. Sequence models are powerful enough to automatically

answer questions, translate between languages, detect emotion, and even generate human-like language. But they are complex and unwieldy. It is too costly to incorporate them into real-world systems and products. To build these models, you need lots of data, skilled human experts, and expensive infrastructure.

This report introduces *transfer learning*, a method of training models that solves these problems through knowledge reuse. Even with transfer learning, these models are still difficult to build. But transfer learning means that, once built, the models can be reused and repurposed by anyone.

Transfer learning and sequence models are not new tools or techniques, but recent breakthroughs use them together. Their combination turns out to be especially powerful. Transfer learning not only improves the accuracy and robustness of sequence models but, more importantly, reduces the cost of using advanced techniques. With transfer learning it is now *practical* to use these techniques.

CHAPTER 2

Background

In this chapter, we will discuss the challenges of language understanding and why they are not easily solved. We will discuss how the field of natural language processing attempts to build models that can understand language, and why this is difficult. Finally, we will introduce transfer learning, a learning technique that greatly reduces barriers to language understanding.

Algorithmic Understanding of Language Is Difficult

Humans process large amounts of natural language data with ease every day. It is central to almost everything we do. In processing all of this data, we seamlessly invoke a number of different complex concepts, rules, and patterns that allow us to understand the *meaning* of natural language, mostly without even being aware of it. Understanding the meaning of a piece of text is easy, but explicitly explaining all of the prior knowledge and reasoning you invoked in order to understand it is much more difficult. Consider the following news excerpt:

"Amazon will offer a free version of its music service, according to a report from Billboard. There are no details about whether it will be on-demand, but Billboard suggest

this new free service will emphasize Amazon as a direct competitor to market leader Spotify."

The meaning of this paragraph no doubt came easily and instantly to you. But you may not realize how many concepts you needed to understand it. For example, you knew that Amazon and Spotify are technology companies, and that "its" in "its music service" referred to Amazon. Likewise, you knew that "it" in "whether it will be on-demand" referred to the music service. You probably could guess that Billboard is a media reporting company based on the context that it had released a report, and that Spotify runs a music service based on the context that this Amazon music service will compete with it. You also may have noticed that the second sentence is grammatically incorrect in American English: the verb "suggest" refers to a single company, "Billboard," and should instead be "suggests."

While it is clear that humans can process language at many granularities, incorporating both syntax and meaning, it is unclear how to equip machine learning models with these same tools—or whether it is even necessary to do so. Do machines need to process language like humans do in order to be effective?

We Need Models That Understand Language

As the field of NLP continues to advance, machines are asked to perform ever more difficult tasks. In this section, we will examine several business applications for NLP and show that they demand advanced language skills. We'll also discuss a subset of those skills in depth, and explore the types of failures that arise without them.

Marketing

Marketing departments need to understand how customers perceive their products or services, and the experiences they have when consuming them. Customers communicate their perceptions and experiences informally by writing reviews, lodging complaints, or asking questions. The language they use can be analyzed to detect positive or negative feelings, which helps companies gauge consumer sentiment. This task is called *sentiment analysis*, and NLP plays a crucial role. Sentiment analysis answers the question, “How does the writer/speaker feel?”

“This is the cleanest hotel I’ve ever seen!” — positive

“Was the room even cleaned after the last guests left?” — negative

Sentiment analysis is conceptually simple, but notoriously difficult in practice. To do well at this task, a model must not only learn words that convey particular sentiments, but also understand concepts like *negation*:

“This car is a pleasure to drive.” — positive

“This car is hardly a pleasure to drive.” — negative

“This car is hardly cheap, but was a pleasure to drive.” — ?

Humans communicate their feelings in varied and sometimes subtle ways. There are simply too many variations to learn all of them. A model for automatic sentiment detection needs to be capable of understanding fundamental concepts like negation, or it will provide misleading results that are unusable at best, and potentially detrimental.

Sentiment models must also understand how context can alter sentiment. Here, the same words convey opposite sentiments just because they are talking about different products:

"This pillow put me right to sleep!" — positive

"This movie put me right to sleep!" — negative

These are short examples, of course. Understanding negation and context becomes more complex with more input text. Basic models for sentiment detection may get it right some of the time, but will humans be comfortable making decisions that affect the future of a company based on "some of the time"?

Routing Customer Inquiries

Who hasn't had an issue with a product or service and been forced to spend their valuable time on the phone, navigating an automated system, only to be sent to a human who then has to reroute them to the correct department? Automating parts of the customer service pipeline is valuable and important, but it can be difficult to get right.

Natural language processing can help analyze customer inquiries and route them automatically to the correct department. Categorizing inquiries by department is essentially the task of *topic classification*. It answers the question, "What is the writer/speaker talking about?"

While topic classification is one of the simpler NLP tasks, these models still need advanced skills to perform well. A simple model might learn to associate certain words with particular departments. For example, "bank" might be associated with the consumer banking department. But these rules are easily fooled:

"I need to make a deposit to my online banking account."

"I was banking on my new credit card arriving today, but it's not here!"

“Microsoft released Q3 earnings on Friday, outpacing other big tech stocks like Pear and Hooli. The tech giant reported an impressive EPS of \$5.28 compared to an average EPS of just \$1.49 for the tech sector.”

Question: “What was Microsoft’s Q3 EPS?”

FIGURE 2.1 *Answering certain questions requires coreference resolution.*

“I just got cut off from a previous chat. I need to be reconnected with the same agent, Susan Banks.”

A model that can differentiate between requests needs to incorporate more than just basic rules: in particular, understanding that a word can be a noun in one case, a verb in another, and a proper noun in yet another. These skills are called *part-of-speech tagging* and *entity detection*, and both require analyzing the surrounding context of words in addition to the words themselves.

Searching Text for Information

Natural language processing can search text for answers to a user’s questions. It does this by understanding the question and what it seeks, and by understanding the text passage (or set of documents) in which to seek the answers. This task

is known as *question answering*. It asks, “What information is the reader looking for?” Question answering requires several different aspects of language understanding. Consider this example:

Answering this question specifically requires the skill of *coreference resolution*, or detecting which references refer to the same entity. The answer in this case is \$5.28, but this EPS is reported as belonging to “the tech giant.” Only by acquiring the skill of coreference can a model correctly identify that “the tech giant” refers to Macrosoft. This simple skill is important for correctly linking facts and references, even when they appear far from each other in the text.

Modeling: Approaches to Representing Language

There are certainly cases where deep understanding of language is not required or provides only marginal benefit, but it is clear that many highly important applications do need tools for processing language in the way humans do. Building models that could potentially understand language at these deeper levels begins with understanding how we represent language.

The first step in building models that can process and reason about language is to make language computable — that is, convert text or speech into a numerical form that a computer can process. Models can then use this numerical form to learn how to map inputs to outputs. Exactly how we represent language is crucial to determining the types of tasks that these models can perform and how sophisticated they can be.

"I loved this movie."



FIGURE 2.2 In a bag-of-words representation, word order is ignored.

Bag of Words

The simplest form of representing text as numerical input is called “bag of words.” With this approach, a piece of text is converted into an unordered collection of word entities.

Representing text this way is simple and fast, but it has severe limitations. In a sentiment analysis example, a model may use this representation to learn that some words have positive or negative connotations. A movie review could then be classified as positive or negative simply by counting the number of positive and negative words and choosing the majority class. This works well in simple cases, but fails in many instances.

Because word order is completely lost with bag-of-words representations, context is lost and it’s difficult to extract meaning. For example, “dog bites man” and “man bites dog”

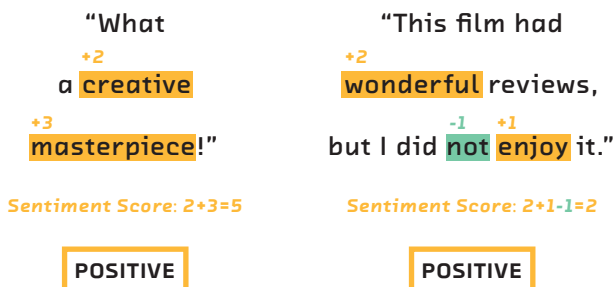


FIGURE 2.3 Basic rules for sentiment analysis work for only the simplest cases.

have the same representation in bag of words, yet they mean very different things. Bag-of-words representations also have no way to preserve synonyms. For example, “magnificent movie” and “splendid film” convey the same meaning but are no more similar to each other than they are to “horrible picture” when using bag of words.

Word Vectors

Instead of treating each word as a distinct symbol, we can choose to represent each word with an *embedding*. Embeddings are continuous representations of words, such that words that have similar meanings will have similar embeddings.

Embeddings are a much better representation for words than a simple bag of words. Phrases can be represented as

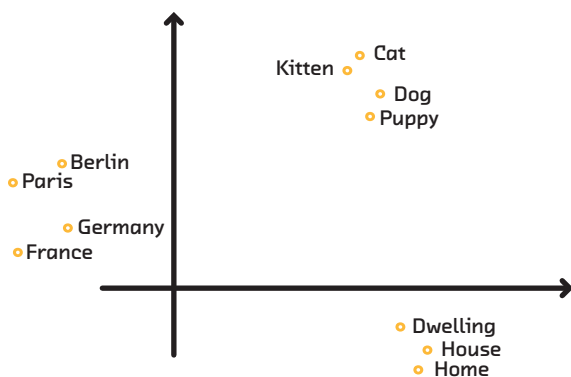


FIGURE 2.4 *Word vector representations can preserve synonyms and even analogies, like “Paris is to France as Berlin is to Germany.”*

sums of word embeddings, so the representations of “magnificent movie” and “splendid film” are now nearly identical.

A model that is learning to map phrases to sentiment labels does not need to see both examples to understand that they are both positive. Since the model only sees their embedded representations, they are more or less the same phrase in the model’s eyes.

Word embeddings alone still have significant limitations, though. They ignore word order, which is crucial to understanding nontrivial cases. Additionally, they do not take surrounding context into account. For example, the word “fall” has a completely different meaning in the phrases “be careful not to slip and fall” and “the leaves are beautiful in the fall.” Word vectors only assign one meaning to a particular symbol,

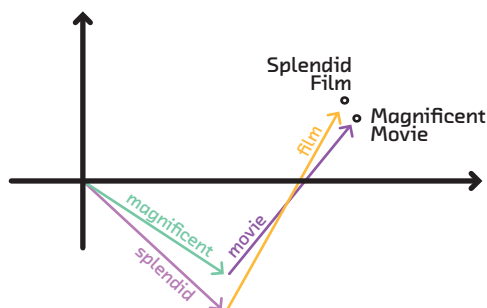


FIGURE 2.5 Word phrases can be represented as sums of word embeddings. “Splendid film” and “magnificent movie” have nearly the same representation, even though they do not share any individual words.

and so cannot use context clues like “careful” and “slip” in order to distinguish the verb “fall” from the noun “fall.”

Sequence Models

If there’s any hope of machines understanding the complexities of human language, they must be able to process language input in a way that preserves the information contained in word order. For example, we know that the following two descriptions mean different things, even though they’re composed of exactly the same words:

“It was great to see this awful movie end.”

“It was awful to see this great movie end.”

There is an entire class of models devoted to processing

language in an order-preserving way, called *sequence models*. Sequence models are capable of processing word vector representations as ordered sequences, using both near and far surrounding context to distinguish the meaning of words or phrases. While word vectors give rich meaning to individual words, sequence models give rich meaning to *compositions* of words. One way of thinking of sequence models is as "contextualizers." That is, sequence models contextualize individual word embeddings, allowing them to change their meanings given the other words that surround them. After all, meaning in natural language is derived not by processing each word in isolation, but by processing compositions of words that make up coherent thoughts, statements, and arguments.

Sequence models are the workhorse of contemporary NLP systems. They are responsible for extracting all of the relevant information from surrounding context and encoding that information in rich, contextualized representations. Predictions for a given NLP task are derived directly from the output of these contextualizers, so the contextualized representations must encode everything necessary to accomplish this task.

Sequence models are powerful enough to understand negation, coreference, syntax and grammar, and other important language concepts. Almost all future research in NLP will process words as sequences, just the way humans process them. But like other models, sequence models learn by example. They require many, many examples to learn effectively, because to a sequence model any difference in word order or phrase length represents a separate data point. Much of the current innovation in NLP, including transfer learning,

involves training sequence models more efficiently - we expect this to be true of future innovation as well.

Practical Challenges to Building Sequence Models

Models that can process language in its raw form and learn its syntactic and semantic nuances have existed for some time. Word vectors and sequence models, each more than a decade old, combine to provide models that can encode many of the subtleties of language. But unfortunately, these models are often difficult and impractical to build.

There are over 170,000 words in the English language. Many of these words have multiple uses, or meanings that change in subtle ways depending on the context. Even with the help of modern word vectors, a model will need to be trained with examples of these variations in order to understand them. In other words, you need massive datasets to train a useful model. Further, learning syntax and grammar from scratch is a challenging and data-hungry task. Even with nearly unlimited data, today's models often struggle to learn the generalizable patterns of language.

And training on massive datasets presents further problems in terms of infrastructure and engineering. High-end compute servers and specialized hardware like graphics processing units (GPUs) are essentially prerequisites for training these types of models. Whether a company owns its own machines or rents them from cloud providers, this infrastructure is expensive.

Even with enough data, proper infrastructure, and an

expressive model,¹ expert-level practitioners need to stitch these pieces together. There is no shortage of complexity. Coordinating the learning process across many machines of varied types is an engineering challenge, while training notoriously finicky machine learning models requires deep understanding of algorithms and programming. It is difficult to find people who have sufficient expertise in these areas, and even when available, they are costly to hire.

What Is Transfer Learning and Why Does It Help?

Machine learning models are trained to perform well on a specific objective, also called a task. Sentiment analysis, entity detection, language translation, and summarization are all examples of different tasks in NLP. Transfer learning is a technique that takes advantage of the fact that these tasks are all related in some way.²

The type of information that a model must learn in order to do well on language translation is likely to also be useful in performing text summarization. Transfer learning enables the reuse of knowledge across related tasks.

Race Car Driving: An Example of Transfer Learning

Consider a toy example where the objective is to train a

¹ “Expressiveness” is a measure of what the model is capable of modeling. For example, sequence models are more expressive than bag-of-words models because sequence models are capable of taking word order into account.

² Transfer learning is conceptually similar to multi-task learning, which we discussed in depth in FF08: Multi-Task Learning, in that both benefit from multiple related tasks. But the two are distinct in that multi-task learning trains a single model to perform multiple tasks at once, while transfer learning trains a common underlying model, then adapts that common model to one specific task at a time.

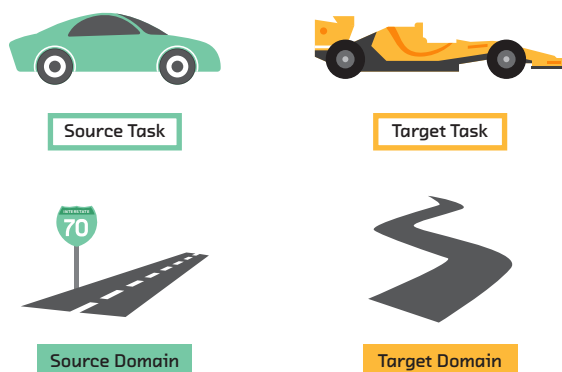


FIGURE 2.6 *Learning a specific target task in a target domain may be difficult, expensive, and time-consuming (like learning to race a Formula 1 car). Transfer learning involves first acquiring related skills on a cheaper, related task (like driving a sedan on public roads). These skills are then transferred to the target task/domain to fast-track learning (a person who can drive a normal car can learn racing more efficiently than someone who has no experience driving at all).*

human to drive a Formula 1 race car on winding professional racetracks. The budget is limited, and it's very expensive to rent a Formula 1 car and purchase time on actual tracks. However, it's very cheap to train someone on the related task of driving a sedan on city streets and interstates. Rather than throw someone who's never driven a car before straight into learning to drive a race car, transfer learning can be used instead.

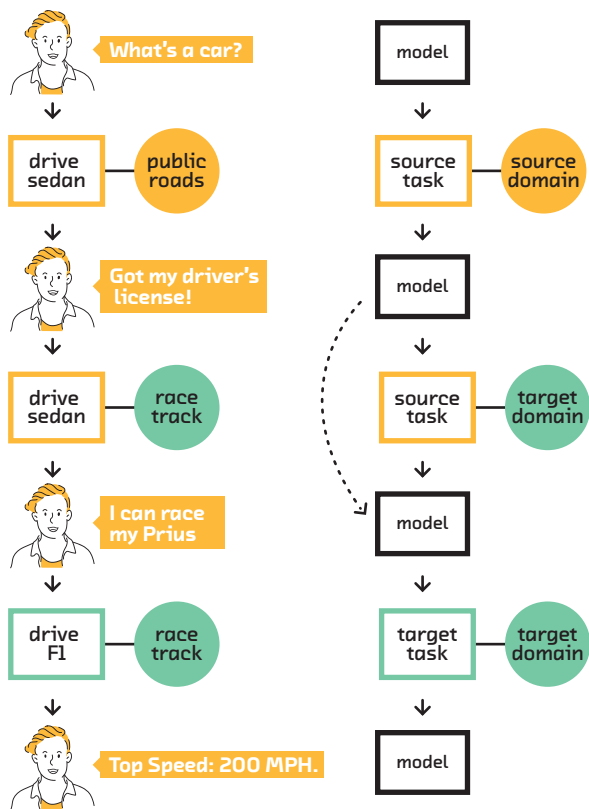


FIGURE 2.7 The transfer learning procedure.

The typical pattern in transfer learning is to learn the bulk of the important knowledge through performing the source task in the source domain. In this example, the human driver may spend many weeks learning to drive the sedan on public roads. Once the driver has acquired adequate knowledge of driving in general, they transfer their knowledge to the target domain. They may optionally choose to spend some time learning to drive the sedan on professional racetracks before switching to driving the Formula 1 car. (This a helpful, but sometimes unnecessary step.) Finally, once the driver is practiced at driving a normal car on difficult racetracks, they learn how to drive the Formula 1 car on the track.

This strategic process allows us to take advantage of tasks or domains that are more readily or cheaply accessible than the target task and domain. In this driving example, it is possible that a human could learn to drive a Formula 1 car on difficult tracks without any prior knowledge, but it would certainly require a lengthy amount of training time using expensive resources. Why would anyone choose to learn the fundamentals of driving in such a specialized situation, when they can easily be learned in a simpler, cheaper setting? Transfer learning helps models master the basics without having access to a lot of fancy, expensive data.

Transfer Learning for Language

Transfer learning is a general, intuitive framework for learning more efficiently. It can be applied to many domains and tasks, and researchers have recently made breakthroughs in its application to language.

Data for a sentiment analysis project might be scarce and



FIGURE 2.8 *Manual labeling of data is often time-consuming and inefficient.*

expensive to acquire, often requiring human labor to annotate examples manually.

With limited data, it would be difficult to train a model that can learn the subtle concepts required to be effective for language tasks like sentiment analysis. However, it is often the case that there is an abundance of data available to learn a related task. One such task is called *language modeling*, which involves predicting the next word given an input prompt.

Crucially, language models can be learned from any corpus of text because the data and labels for a language model come from the text itself. Knowledge learned through language modeling is often useful in other tasks. For example, predicting the next word requires mastering the concept of negation, which as we've seen is also important for performing sentiment analysis.



FIGURE 2.9 *Language models predict the most likely next word for a given prompt.*

Transfer learning takes advantage of the fact that the skills required to do these two tasks are related. The target task—sentiment analysis, in this example—requires labeled data, which is scarce and expensive to acquire. The source task—language modeling—has an abundance of data available because no labels are required. Therefore, transfer learning chooses to acquire common skills using the cheaper source task and domain, and then applies those skills to the target task and domain. In general, transfer learning works better when the source and target domains are more similar and when there is high overlap of skills for the target and source tasks.

What Does Transfer Learning Get Us?

Transfer learning means that models no longer start

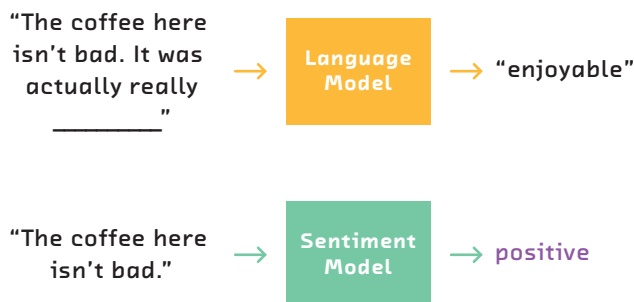


FIGURE 2.10 *Both language models and sentiment models need to learn negation to be accurate.*

learning from zero. Crucially, the starting models—that is, models pretrained on the source task in the source domain—can be acquired for free in many cases. Many state-of-the-art pretrained models are published openly and can be used by anyone. In some cases, using these pretrained models simply makes the training of the target task faster, which in turn saves on costly resources. In other cases, the model can start out with knowledge crucial to performing the target task that it would not otherwise have been possible to acquire, either because there is not enough target data or because it is too expensive to acquire more. Even when there is enough data for the target task, it can be difficult for practitioners who do not have deep expertise in the field to get the model to learn the right things. Beginning with a pretrained model

greatly reduces the complexity of the training process, making it more manageable for the average practitioner.

CHAPTER 3

Technical

In this chapter, we'll define the basic structure of modern NLP models and present recent breakthroughs in transfer learning in the context of that structure. We'll then discuss practical steps for how to implement transfer learning and when it's appropriate.

The Structure of NLP Models

Though modern machine learning models for text processing can be quite complex, each system can be broken down into parts that are simple to understand. Each of these systems maps an input (text) to some sort of prediction — like a translation, a sentiment classification, or a category. Nearly all modern NLP systems perform this mapping by passing the input text through the same series of basic subcomponents. Understanding these subcomponents is helpful in making sense of the latest techniques in transfer learning.

Each subcomponent has a simple purpose, though it may achieve that purpose in complex ways.

In the *tokenization* phase, the input text is converted into an ordered sequence of individual units called *tokens*. Tokens are usually words ("loved"), but could also be word pieces ("lo",

"ved") or even individual characters ("l", "o", "v", "e", "d").³ The model that processes the text needs to understand the meanings of these tokens and how they can be composed.

The *token embedder* turns a sequence of tokens into a sequence of *token embeddings*. Token embeddings are numerical (specifically, vector) representations of tokens. Ideally, these embeddings will faithfully represent the meaning of the tokens. That is, similar tokens (such as synonyms) should have similar token embeddings. One important aspect of token embeddings is that the embedding for a given token depends only on the token itself: a particular word will always map to the same token embedding, regardless of the surrounding words.

The *contextual token embedder* (or simply *contextualizer*) converts this sequence of isolated token embeddings into a sequence of contextualized token embeddings. That is, after the contextualization stage, the embedding for a particular token has taken into account the surrounding words. For example, the word “right” will have a different contextualized embedding in “turn right on Main Street” vs. “you’re right on the money.”

In the *prediction head* phase, the contextualized sequence representation is mapped to a prediction—which could be another sequence, a vector, a number, or a category, depending on the specific task.

During training, only the token embedder, contextualizer, and prediction head modules are learned. With transfer learning, instead of training each of these modules from

³ For clarity, in most places in this chapter we will assume words are used as tokens, but it should be noted that other types of tokens are possible and common.

scratch (i.e., randomly initializing them), at least one of the modules is initialized from its counterpart in a model trained on a different task or domain. For example, word embeddings learned from a dataset of movie reviews could be “transferred” or reused in an analysis of restaurant reviews, since the word meanings are likely to be very similar. In fact, it has long been common practice to transfer token (word) embeddings. Only recently has the transfer of the contextualizer been explored.

Transferring Token Embeddings

The most basic way of representing text is to view it as a sequence of symbols, where each word is a separate symbol. With enough data, it is possible to learn what these symbols mean. But as we discussed in [2 Background](#), this representation is inefficient. Instead, representing each symbol with a continuous embedding (think of it like a “meaning vector”) allows us to take advantage of the inherent structure of language, preserving the information contained in, for example, synonyms and analogies.

The basic idea behind word embeddings has existed since at least 1986, and they were first used to represent text in deep neural networks⁴ in 2003. A well-known paper written in 2013⁵ introduced *Word2Vec*, a procedure that enabled learning word embeddings at massive scales. Using this algorithm and its successors, it’s possible to take any corpus of text and generate a set of rich “meaning vectors” that offer huge improvements over representing words as discrete symbols.

With the breakthroughs that enabled learning word

⁴ <https://dl.acm.org/citation.cfm?id=944966>

⁵ <https://arxiv.org/abs/1301.3781>

1. Train a token embedder using Word2Vec



2. Transfer that token embedder to other models

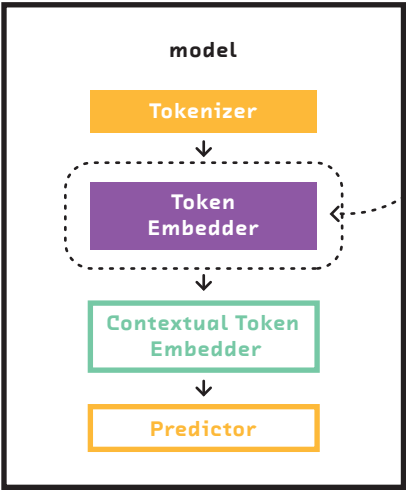


FIGURE 3.2 Token embeddings can be learned from a large corpus like Wikipedia, then transferred to any NLP system that leverages token embeddings.

embeddings at these scales came a new realization: word embeddings learned from a sufficiently large and general dataset could be reused in learning tasks on different datasets. While natural language datasets may differ significantly, the words used in those datasets and their meanings are still roughly the same.

Reusing word embeddings across different tasks and domains is a form of transfer learning, and has been common practice in cutting-edge NLP models for many years. Without the ability to transfer word embeddings, certain small datasets would not be sufficient to learn word meanings from scratch, and we would be forced to use the simpler approach of treating each word as a distinct symbol. Even when the training data is sufficient, it is still a waste of resources to relearn word meanings for every new dataset.

Transferring Contextual Token Embeddings

Token embeddings are powerful because they take advantage of *knowledge transfer*. This reduces the complexity of training a model because it eliminates what must be learned from the target dataset. But the token embedding stage is just one part of the model as a whole; the contextualization stage, which is generally more difficult to learn, often has a greater impact on the model's performance. If the token embedding stage is responsible for learning the meaning of words, the contextualizer is responsible for learning the meaning of language. It may need to extract many different types of information.

It's natural to wonder if it's feasible to perform knowledge transfer at the contextualization level. For this to be possible,

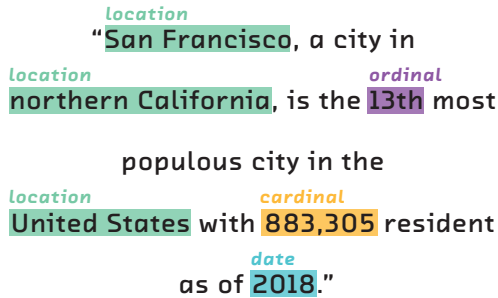
NEGATION

"This movie **wasn't** exactly my favorite."



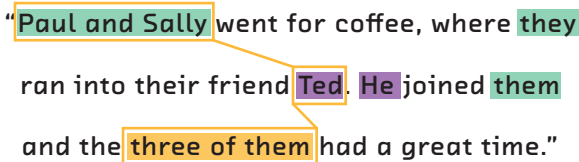
ENTITY RECOGNITION

location
"San Francisco", a city in
location northern California, is the *ordinal* 13th most
populous city in the
location United States with *cardinal* 883,305 resident
as of *date* 2018."



COREFERENCE RESOLUTION

"Paul and Sally" went for coffee, where they
ran into their friend Ted. He joined them
and the three of them had a great time."



DEPENDENCY PARSING

"John" tapped his fingers while
listening to the music.

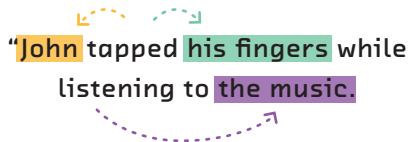
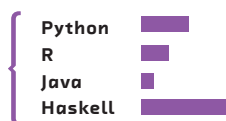


FIGURE 3.3 The contextualizer is a skilled language processor. It may learn how to perform various useful language tasks like recognizing negation, picking out entities, identifying coreferences, and identifying dependency structures.

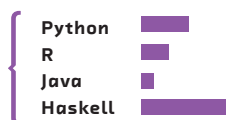
FORWARD LANGUAGE MODEL

The best programming language is _____.



BACKWARD LANGUAGE MODEL

_____ is the best programming language.



MASKED LANGUAGE MODEL

The best programming _____ is Python but _____ is pretty good too.

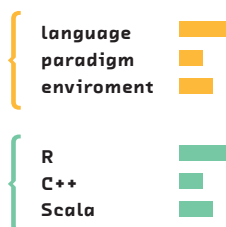


FIGURE 3.4 Language models may predict missing words using previous words, future words, or both.

there must be some other source language task that teaches the contextualizer how to extract useful pieces of information from text. The information it extracts should be generalizable — not too specific to any one dataset — so that it can be transferred across many domains. Additionally, there should be plentiful data with which to teach the contextualizer.

Language models learn to predict missing words, given a sequence of input words. This task is "self-supervised," since the labels for each example come from the data itself. This means that any collection of text could be used to train a language model — no human labeling is required (though there are trade-offs in choosing the data for a language model, which we discuss in [3.4 How Do You Actually Use Transfer Learning?](#)). It turns out, however, that predicting missing words is a difficult task that requires a deep knowledge of language. Therefore, the process of training a good language model forces the contextualizer to learn many generic language concepts. Such a contextualizer will also be useful on other tasks and in other domains.

The idea of transfer learning for NLP is that we can invest a lot of resources into creating one really "smart" model that can then be reused over and over, ad infinitum. This model could then be adapted for use in other NLP applications, which would take advantage of the model's starting knowledge. For a long time, though, it was not clear if a single model could encode enough language understanding to be useful, and even if this were possible, it was not obvious what training process would enable it. It was also assumed that massive

datasets would surely be required. Research has revealed language modeling to be the answer.

The recent success of language models in transfer learning is undeniable. Because language modeling doesn't require labels, data availability is not a problem. But still, these language models do require massive datasets and are incredibly expensive to train. This—in part—explains why this discovery wasn't made sooner; it was hard to justify the investment to train a language model when no one was sure it would work. Now it is apparent that language modeling does yield a contextualizer that can encode diverse, general features of language that are useful in many related tasks.

It is not easy to say what, exactly, these models have learned. There have been several interesting research projects that aim to answer this question. Research from the Allen Institute for AI showed that these pretrained contextualizers learn different language concepts at the different layers. For example, the earlier layers learn to extract local structure, like part-of-speech tagging. In contrast, the later layers learn longer-range relationships, like resolving coreferences. Researchers at OpenAI⁶ showed that contextualizers trained on language modeling tasks automatically learn other skills like reading comprehension, summarization, and translation. In contrast to traditional multi-task learning, which we discuss in detail in

⁶ <https://openai.com/>

FFo8: Multi-Task Learning⁷, these contextualizers are not trained explicitly to perform multiple tasks. They are implicit multi-task learners because they learn to do these things simply because it makes them better at the task they are asked to perform: language modeling.

The ability of contextualizers to learn both short- and long-range relationships by training on just language modeling explains why they are successful in the transfer learning setting. The pretrained contextualizers learn a broad set of language skills, and downstream tasks often require one or more of these skills. The process of transfer learning enables a generalist contextualizer to specialize in a particular target skill, outperforming a contextualizer trained from scratch.

The ability to transfer knowledge at both the token embedding and the contextualization levels is extremely powerful. Most NLP tasks can now begin with the token embedder and the contextualizer already trained. All that's left is to train the task-specific predictor to output the right kinds of predictions. This obviously saves time and money, but there are also other benefits. The process is much simpler, meaning that it doesn't take a data scientist with years of experience to train a useful model. And beyond the fact that the contextualizer and token embedder are already trained, they are almost always smarter than they would be if trained on just the target dataset. Transfer learning yields more accurate results, faster, and at a lower cost.

⁷ <https://clients.fastforwardlabs.com/ff08/report>

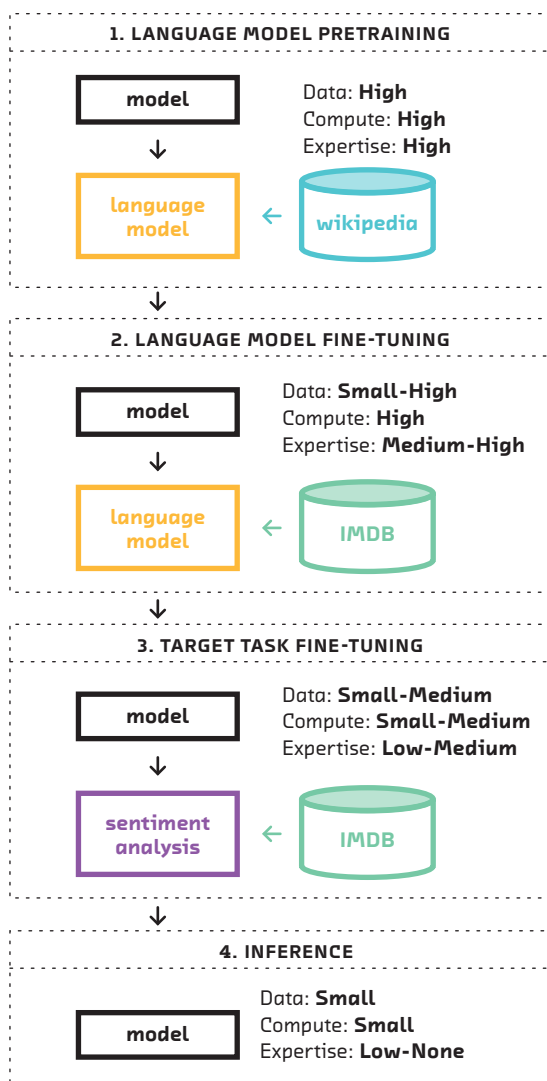


FIGURE 3.5 The steps of applying transfer learning.

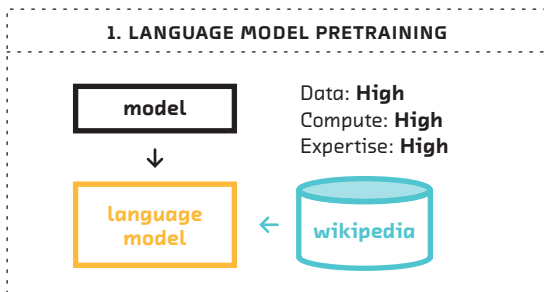
How Do You Actually Use Transfer Learning?

Transfer learning is a very general idea and so can be implemented in various ways, but for the sake of clarity, we will define a specific step-by-step process as a reference. This process is extremely modular. You may choose to incorporate as many or as few steps of the process as you like, trading off accuracy for development cost. Because every organization has different needs and resources, it makes sense to understand what each of these steps involves and how you might incorporate them. The example we'll walk through shows transfer learning as applied to sentiment analysis.

Step 1: Teaching the contextualizer generic language skills

In this step a contextualizer is trained as a language model on a giant corpus of text. A common example is training using all of Wikipedia, which covers a diverse and wide-ranging set of topics. Because this step requires training a large contextualization model from scratch, it is difficult and costly. Most practitioners skip this step and allow others to do it for them.

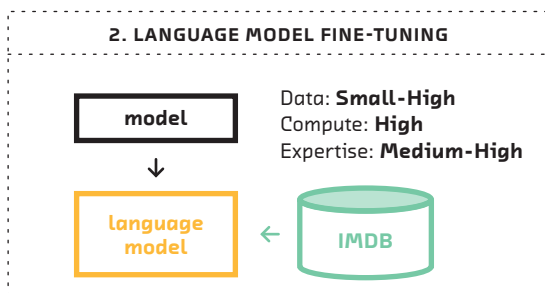
Large organizations like Google, OpenAI, and other research labs have openly published their models so that the public can use them. Most of the time, it will suffice to simply download a pretrained model from the internet and plug it into one of the later steps. In cases where the target domain may involve highly irregular language, however (for example, shorthand found in doctors' notes), standard off-the-shelf models may be of limited use. In this case, there may be pretrained models available for the target domain (for example,



a version of BERT which specializes in biomedical language, which we discuss in [5 Landscape](#)), or this step could be performed manually.

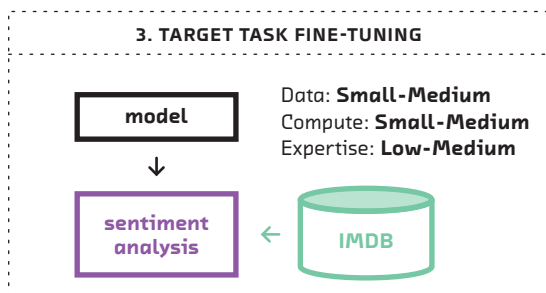
Step 2: Fine-tuning the contextualizer

This step is also optional and is often skipped. It involves tuning the contextualizer to do language modeling on the target dataset. This step might be applied when the target dataset is *somewhat* different from the dataset used in the first step. For example, when analyzing tweets, *most* of what a model learned on — say, Wikipedia — may still be beneficial, but it may be necessary to refine that model to learn, for example, various types of slang common on Twitter. Relatively little is known about when it is necessary to include this step. Because it can be difficult to implement, we recommend skipping this step as a default, and especially when using tools that don't provide proper support for it.



Step 3: Fine-tuning the target task

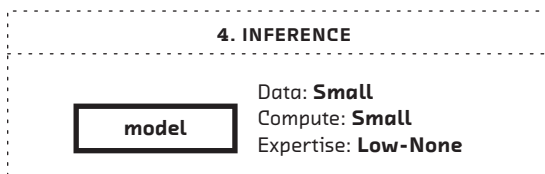
This step is included far more often than the first two steps. Using the output of step 1 or step 2 as a starting point, the contextualizer is fine-tuned or refined using the target dataset for the target task. The contextualizer has already learned how to extract concepts from language, but may benefit from honing those skills for a particular task, like sentiment analysis. This step may also allow the contextualizer to "forget" things that may be irrelevant (e.g., the ability to detect entities may not be important for sentiment analysis), and instead use that capacity to learn new concepts. The final accuracy of the model almost always benefits from this step, since there are usually important nuances in the target data that could not be learned from the source dataset. The decision of whether to include it or not simply involves balancing the trade-offs between complexity and accuracy for a particular application.



Step 4: Inference

The final step involves taking the model from the output of step 1, 2, or 3 and using it to contextualize the input text. While steps 2 and 3 do require extra training, there are plenty of freely available pretrained models (which we discuss in [3.5 Model Variants](#)) that have been trained according to step 1. Since it is always possible to use those models directly, no extra training is required.

From a functional perspective, it does not matter where the contextualizer came from. There are simply performance trade-offs to consider. The outputs of the contextualizer are contextualized word or token embeddings, which can be plugged into many downstream models—and not just deep learning models. For example, it is possible to download a state-of-the-art pretrained model from the internet, use it to contextualize an entire dataset, and then plug those



contextualized tokens into a support vector machine for prediction. No deep learning is required.

In practice, we find that it is rarely sufficient to perform only this final step. Allowing the pretrained models to further specialize through fine-tuning usually provides large gains in accuracy.

The process of transfer learning is extremely flexible, with many options and trade-offs to consider. Next, we discuss some general guidelines on when to use — or not use — transfer learning.

When Is Transfer Learning Helpful

Use transfer learning when:

- Context is important to the task.
- State-of-the-art performance is important but there are constraints on talent, time, or money.
- Black box models can be tolerated or tools are available to

add interpretability.

- The complexity and latency of deploying a large neural network are acceptable.

Transfer learning for NLP promises to equip any model for any task with the ability to understand language, at low cost. But there is a cost. Even in the simplest case, there is a cost to deploying a large, complex neural network model, which could become a bottleneck if not done properly. Further, using these pretrained neural networks means accepting their black box nature. Decisions and predictions will be more accurate, but there will be a lack of trust in the model—a topic we discuss at length in our report on Interpretability⁸ in machine learning. Understanding why and how transfer learning works will provide insight into when its benefits may outweigh the costs.

Following recent technical breakthroughs, it is now possible to transfer the contextualizer stage of an NLP model. This means that transfer learning will be most beneficial when it is important to have a sophisticated contextualizer. It is difficult to say exactly what a contextualizer knows or how it works, but we have already discussed what the contextualizer can do. One way of thinking about it is to say that the contextualizer “learns language.” So, as a general rule, transfer learning will only be beneficial for tasks where understanding language is beneficial.

One example of where learning language is not always important is in topic classification—the task of classifying text into categories. It is possible to reach near-maximum performance on this task using simple heuristics, like mapping

⁸ <https://clients.fastforwardlabs.com/ff06/report>

specific words to specific topics. On the other hand, a task like question answering requires understanding language in several ways. A contextualizer is important for identifying entities, understanding word and sentence ordering, and picking out dependency structures in the text. Question answering would surely benefit from transfer learning. Sentiment detection lies somewhere in between. Simply mapping words as either positive or negative would succeed in some cases, but sentiment can be varied and subtle. Nuanced cases will require a deeper understanding of language.

Nothing really beats the “try it and see” approach, but contemplating the types of information that a human would have to leverage to perform such a task is a great barometer. If context is important to the task, then transfer learning will likely help.

Model Variants

Up until this point, we’ve discussed the history, theory, and procedure of transferring pretrained models, but we haven’t discussed any specific models, how they are trained, or exactly where to find them. In order to actually put transfer learning into practice, you’ll need to choose a particular model architecture and obtain a pretrained version of it. There’s no shortage of options. At the time of this writing, there are new model variants being created and published every few weeks. We expect that trend to continue for the foreseeable future, so the best options will change quickly over time.

Still, it’s worth exploring some of the currently available options, especially since these are the models that launched the field. Each of these models can be used for transfer

learning at the contextualization level, because they all produce a contextualizer that takes in a sequence of words and produces contextualized word embeddings. They differ in model architecture, model size, how they were trained, and what data they were trained on.

We'll discuss several of the foundational models for transferring contextualizers in NLP models. It is important to note that each of these works is roughly defined by its choice of pretraining task (how the contextualizer was taught) and the model's architecture (what kind of neural network the model uses). These combinations can be mixed and matched with various trade-offs, and indeed much of the current research is devoted to exploring new combinations.

The following models are based on one of two basic architectures for the contextualization module: the *recurrent neural network* (RNN) or the *transformer*. The specific differences between them are interesting, but the details are highly technical. We discuss RNNs in detail in our report on Summarization⁹. The details of the transformer architecture are beyond the scope of this report, but we find The Illustrated Transformer¹⁰ to be a great resource for the technically curious.

ELMo

Architecture: RNN

Pretraining task: *Bidirectional language modeling*

⁹ <https://clients.fastforwardlabs.com/ff04/report>

¹⁰ <http://jalammar.github.io/illustrated-transformer/>

Tokens: *Words and characters*

Pretraining data: *1B Word Benchmark*

Available from: *AllenNLP, TensorFlow Hub, GluonNLP, Chainer*

ELMo stands for Embeddings from Language Models. This was one of the first models to enable transfer learning at the contextualization level. It is based on an RNN architecture, which is just a type of neural network model that enables processing data as a sequence, like text. ELMo was trained on a language modeling task, where it attempts to predict the next words in a given input sentence across thousands of examples.

One drawback with the ELMo model is that the model was trained on *shuffled* sentences, which means that ELMo was not trained to learn long-term dependencies. So if you'd like to use transfer learning in a task that requires incorporating context across sentences, ELMo may not be a good choice. ELMo has easy-to-use public implementations in the AllenNLP library, which we discuss in [5 Landscape](#).

ULMFiT

Architecture: *RNN*

Pretraining task: *Language modeling*

Tokens: *Words*

Pretraining data: *Wikitext-103*

Available from: *Fast.AI*

ULMFiT, which stands for Universal Language Model Fine-Tuning, defines more of a particular *process* for transfer learning than a specific architecture. It stands out from some of the others in this category because of its emphasis on the language model fine-tuning step, which we discussed in [3.4.1 Step 1: Teaching the contextualizer generic language skills](#).

In ULMFiT, an RNN is first trained as a language model, similar to in ELMo.¹¹ However, in this case there is an intermediate step where the trained contextualizer from the first step is trained again as a language model on the target corpus. This is exactly step 2 from [3.4.2 Step 2: Fine-tuning the contextualizer](#). This step helps adapt the contextualizer's model to the target domain before it is deployed for whatever the target task is.

ULMFiT has been shown to work particularly well for a variety of text classification tasks—an application that is extremely common, but is not as well studied as some other tasks like machine translation or question answering. ULMFiT has public implementations available from the Fast.AI library.

GPT

Architecture: *Transformer*

Pretraining task: *Language modeling*

Tokens: *Word pieces*

Pretraining data: *BooksCorpus*

Available from: *AllenNLP, Huggingface, GluonNLP, Chainer*

In 2018 research lab OpenAI released a model based on generative pretraining (GPT) that built on the ideas from previous transfer learning models and applied them to a transformer architecture instead of an RNN. A transformer model is pretrained to do next-word prediction, then used as a contextualizer in several other language tasks. This particular

¹¹ Because ULMFiT is a process for transfer learning it can actually be applied to various types of model architectures, but most public implementations and studies have been done with RNNs.

work is interesting because it showed that the transfer learning paradigm for NLP worked well with other architectures besides RNNs, and began a trend of transformer-based pre-trained models.

GPT has since been superseded by a new version (GPT-2), which is a larger model trained on significantly more data. Because these models were pretrained as true language models, they are capable of *generating* text as well. The GPT-2 model is so good at generating human-like text that the largest and most powerful version of it has not been publicly released at the time of this writing, due to the authors' stated fear of potential misuse for malicious purposes.

BERT

Architecture: *Transformer*

Pretraining task: *Masked language modeling and sentence-pair classification*

Tokens: *Word pieces*

Pretraining data: *BooksCorpus, Wikipedia*

Available from: *AllenNLP, Huggingface, TensorFlow Hub, GluonNLP, Chainer*

BERT (Bidirectional Encoder Representations from Transformers) uses a transformer model architecture and a modified pretraining task that presents several new and interesting ideas. The creators of BERT argued that other transfer learning models like ELMo and GPT were not as effective as they could be, because they only used unidirectional context—yet evaluating a word and its meaning depends not only on past context or future context, but on both. Other models were limiting in that they made next-word predictions based only

on looking before or after the word they were predicting, but never both; bidirectionality, BERT's creators argued, was the key.

The BERT model was therefore trained as a masked language model (which we discuss in [3.3 Transferring Contextual Token Embeddings](#)). In this setup, certain words in an input sentence are hidden, and the model is asked to predict their identities by incorporating both forward and backward context. BERT also adds a second pretraining objective, which involves predicting whether one sentence is the successor of another, or if they are unrelated:

“San Francisco is a city in northern California.” “With a population of 883,305, it is the fourth most populous city in California.” – True

“San Francisco is a city in northern California.” “The giant panda is a bear native to south central China.” – False

BERT was shown to outperform its predecessors by significant margins across many different language tasks. Since the only significant difference between BERT and other models is its pretraining objective, this effectively shows that carefully choosing the pretraining objective can provide powerful advantages.

CHAPTER 4

Prototype

Perhaps the most exciting aspect of transfer learning is that it is so broadly applicable. It does not enable one specific capability, like question answering, language translation, text classification, or search. It makes *each* of these more accurate, at a lower cost, with less data, and thus more accessible to a broader community. It opens the pathway from research to production.

We built a prototype, Textflix, that leverages transfer learning for sentiment analysis. Textflix performs positive or negative sentiment detection on movie reviews, which present a challenge to text processing systems because of the complex ways in which humans express their preferences. We added LIME, an interpretability technique discussed in FFO6: Interpretability, to provide insight into the model's predictions and used this mechanism to provide individual summaries of each movie. The entire product was built using a model trained on just 500 labeled examples. The modeling was implemented with off-the-shelf tools from AllenNLP and did not require writing any code. Everything was completed within an infrastructure budget of \$25.

Sentiment

While transfer learning is likely to be beneficial for almost

any NLP application, some will benefit more than others. Even though transfer learning can be implemented at low cost, we do not recommend using a sophisticated deep learning model for a 1% gain in accuracy. Some applications, like basic topic categorization, can be handled admirably by extremely simple statistical models (like Naive Bayes, which is essentially counting words). In such cases, sticking with the simple model works best.

But sentiment detection—the task of inferring how the writer or speaker feels—requires sophisticated tools. When humans convey their feelings, they may use sarcasm and irony. They might incorporate obscure pop culture references. They might have mixed feelings, contradict themselves, or even change their minds by the time they’ve finished their thoughts! Without transfer learning, all of these difficult challenges would need to be solved by learning from a single, possibly small, dataset. Because of these difficulties, sentiment detection demonstrates very clearly the power of transfer learning.

Dataset

Textflix is built on the IMDB dataset, a popular open dataset commonly used for sentiment analysis. The dataset consists of 25,000 movie reviews from the users of the popular online movie database, although only 500 of those were used to build the model behind Textflix, as a demonstration of the power of transfer learning.

While sentiment analysis can be applied in many different domains, movie reviews are interesting because they present a diverse set of challenges. Many of the reviews are

straightforward and simple, but some contain subtle clues as to the author’s opinion. These subtleties are challenging for machine learning models—a model based on simple statistics will not work well. Because the reviews are written in rather plain English (no specialized dialects or slang), publicly available pretrained models (which were trained on generic English) will work well.

Models

The modeling process for Textflix was extremely simple. One of the great benefits of transfer learning is that it eliminates the need to invent complex new neural network architectures that uniquely solve a particular problem. Off-the-shelf transfer learning models already provide state-of-the-art accuracy, so the fewer changes we make, the better. After comparing several models, we ended up using the large version of the BERT model (which we discuss in [3.5.4 BERT](#)).

We experimented with several popular transfer learning models, and also compared their performance to simple but strong baseline methods. Although we built Textflix with a model trained on only 500 examples, for each model we explored its performance curve when training on more labeled examples. In the following sections we present these results and justify our modeling choices.

Baseline Models

Testing baseline methods is an important first step in the modeling process. Even if they are unlikely to yield a usable model, they are easy to implement, present a logical reference for comparing future models to, and are usually more

interpretable than more sophisticated alternatives. In some cases, the simplicity and interpretability advantages they present may outweigh the decrease in accuracy in the results they produce. For our prototype, we explored two baselines: SVM with Naive Bayes features NB-SVM¹² and word vectors.

NB-SVM

For text classification problems like sentiment analysis it makes sense to choose a simple model based on bag-of-words as the first baseline. In many text classification problems, like topic classification, these types of baseline models may even be the best choice. NB-SVM treats the text as a bag of words and combines a Naive Bayes model (also a reasonable baseline) with a support vector machine. This model has been shown to produce strong linear baselines for text classification, and sentiment analysis in particular.

In testing this model we generated both uni- and bigram features for the NB-SVM classifier, removed stopwords from the input, and used a Snowball stemmer¹³ to normalize each word.

The performance of the NB-SVM was poor and unpredictable at low dataset sizes—it simply did not have enough observations to learn which words strongly correlated with positive or negative sentiment. However, at larger training set sizes (e.g., 10,000 examples), this baseline reached a useful accuracy of about 85% with very little tuning.

This performance curve shows that even simple bag-of-words models can identify a large majority of sentiment

¹² <https://aclweb.org/anthology/papers/P/P12/P12-2018/>

¹³ <https://snowballstem.org/>

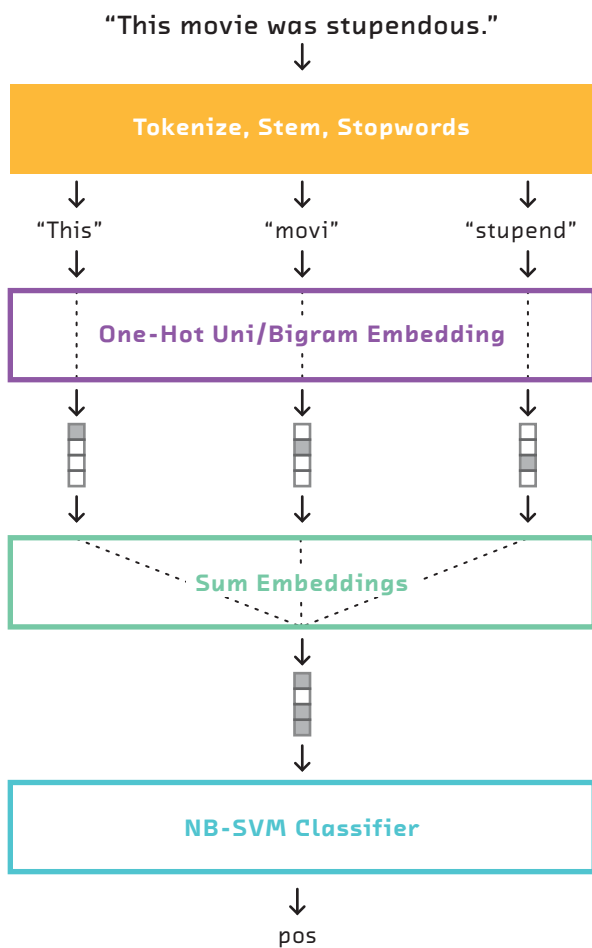


FIGURE 4.1 Pipeline architecture for the NB-SVM model.

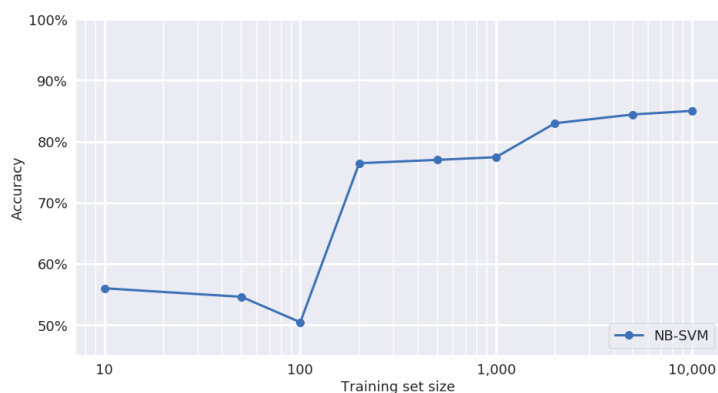


FIGURE 4.2 *NB-SVM is no better than guessing at training set sizes < 200.*

examples correctly. And because the NB-SVM model is simple and fast to implement, it would be easier to support in a production use case than some deep neural network models.

Word Vectors

As we've seen, the NB-SVM model cannot be reasonably expected to perform well with small training datasets. Because it is a bag-of-words model, it has no a priori knowledge of words and their associations and must learn everything from the training data. A model that leverages word vectors should do better in the small-data regime, since word vectors are a form of transfer learning. That is, the meaning of words is already captured in the pretrained word vectors, and is not affected by the small data size.

We used the following simple architecture for this model.

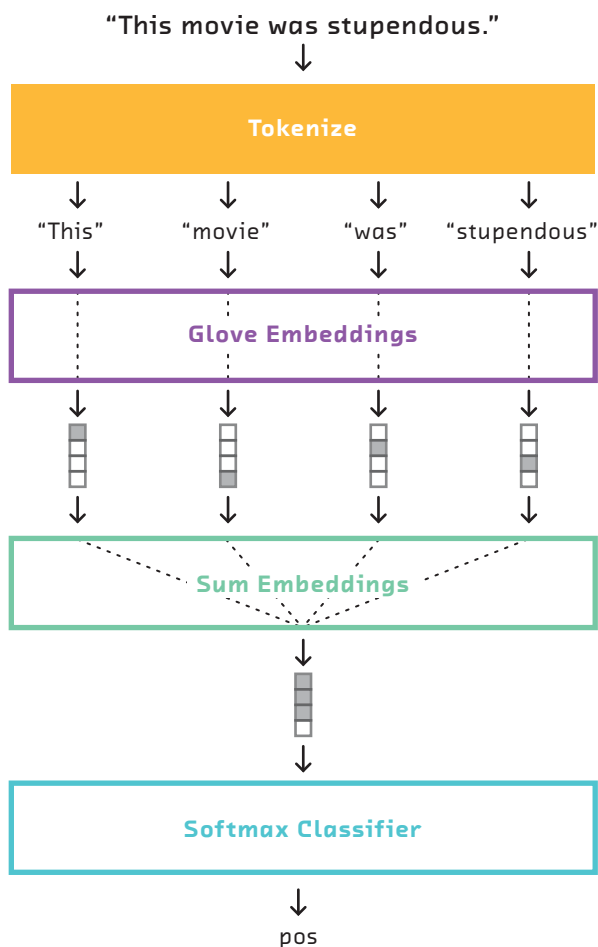


FIGURE 4.3 Pipeline architecture for the word vectors model.

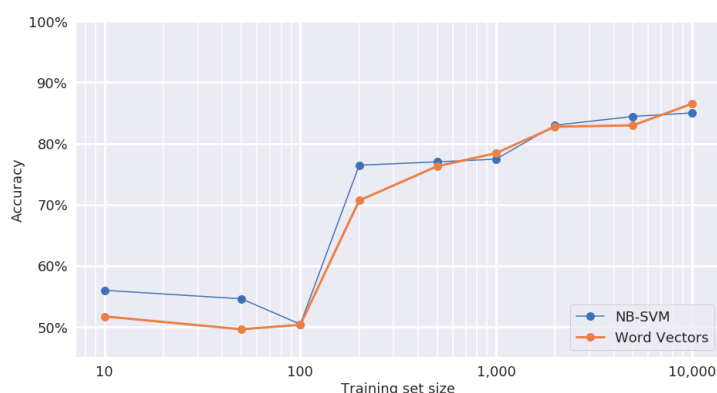


FIGURE 4.4 *Transferred word vectors suffer from the same problems as the NB-SVM model.*

The performance curve shows that the word vectors model generally follows the same trend as the NB-SVM baseline. (It is important to note that there is a lot of variance at each data point, which is not shown in the plot, so small differences should be taken lightly.)

This result is, in some ways, surprising since word vectors are a form of transfer learning and should therefore be more resilient to limited data than the baseline NB-SVM model. It is likely that the word vectors model could be tuned to outperform NB-SVM at smaller training sizes by adjusting the hyperparameters and architecture. However, we purposely spent little time optimizing hyperparameters with any of the models. The hyperparameter optimization step is largely a heuristic process and can require deep expertise to guide the search — something we wanted to avoid.

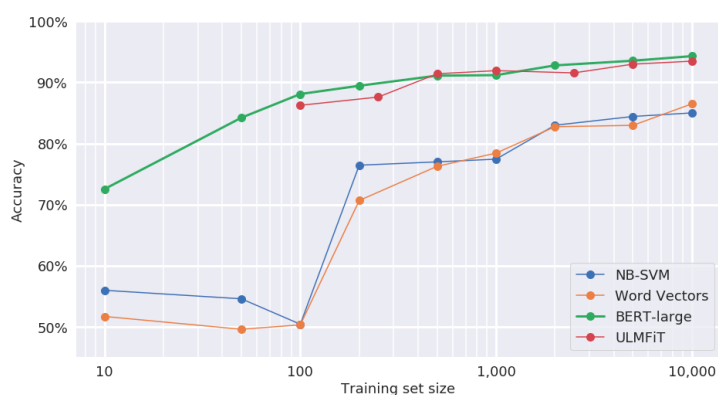


FIGURE 4.5 *New transfer learning models can perform well even with fewer than 100 examples.*

Transfer Learning Models

With well-performing baselines established, we began to try out several of the publicly available transfer learning models. We experimented with ULMFiT using the Fast.AI library and found it to perform well, even with limited data. We performed language-model fine-tuning using 50,000 unlabeled examples for the ULMFiT model, and then performed supervised training at various dataset sizes. In addition to ULMFiT, we ran experiments for both the BERT-Base and BERT-Large models, using the AllenNLP library.

The best-performing models were the BERT-Large and ULMFiT models. While these models produced nearly equal results, we found that the BERT-Large model was easier to implement and experiment with. This is in part because no language model fine-tuning step was required with BERT-Large

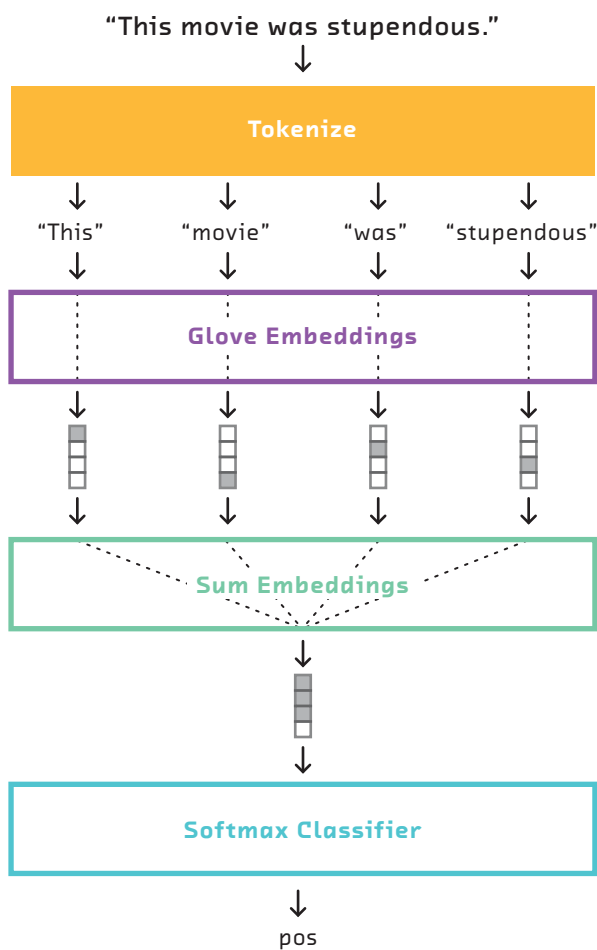


FIGURE 4.6 Pipeline architecture for the BERT-large model.

(target task fine-tuning, however, was performed for each model), and in part because ULMFiT is trained via the Fast.AI library, which imposes development via notebooks—something we find to be restrictive.

We trained the BERT-Large model without making any custom modifications and defined the entire experiment in a JSON configuration file in the AllenNLP style. We trained on a single GPU for 20 epochs, using an Adam optimizer, and used gradual unfreezing for fine-tuning the layers of the model.

Overall, we did very little tuning of the model's hyperparameters. This limited tuning requirement is one of the greatest benefits of transfer learning: the out-of-the-box performance is already very good, and eking out a further 1-2% of accuracy has diminishing returns and would require expensive-to-acquire knowledge of the model architecture.

The final BERT model provides accuracy roughly equivalent to the state-of-the-art model, using just 500 labeled examples for training.

Interpretability

We discussed the costs of transfer learning in [3.4.5 When Is Transfer Learning Helpful](#), and one of those costs was the black box nature of neural network models. The ability to detect sentiment is undeniably useful, but the ability to explain those predictions is significantly more powerful. With *interpretability*, we can not only classify the user's sentiment but point to specific evidence to support that classification. We found the addition of interpretability to the model's predictions to be surprisingly useful. Here we will discuss our approach and some of the benefits.

classification: positive · 82.3% certainty

by consequential_countdown · 143 days ago

So, you wanna be a rock star? See this movie. You don't like rock, you say? Or you're REALLY into heavy metal? Then put on your favorite album and dream yourself away, this movie has nothing to offer. Rarely have I ever seen a movie being able to portrait the dream of being in a rock band as good as this. I had long hair during the late 1980's and early nineties, and I have played guitar for the last 15 years or so. Did I like Rock Star? Oh yes. The music is good, not great, the actors are good, and believable, even Jennifer Aniston plays her part to perfection. And Mark Wahlberg is perfect as the wannabe rock singer. So you know what you're going to get. A movie about dreams coming true, being stepped on, and finally figuring out what life is really about. It's a good solid seven out of ten, no more, no less.

classification: positive · 82.3% certainty

by consequential_countdown · 143 days ago

So, you wanna be a rock star? See this movie. You don't like rock, you say? Or you're REALLY into heavy metal? Then put on your favorite album and dream yourself away, this movie has nothing to offer. Rarely have I ever seen a movie being able to portrait the dream of being in a rock band as good as this. I had long hair during the late 1980's and early nineties, and I have played guitar for the last 15 years or so. Did I like Rock Star? Oh yes. The music is good, not great, the actors are good, and believable, even Jennifer Aniston plays her part to perfection. And Mark Wahlberg is perfect as the wannabe rock singer. So you know what you're going to get. A movie about dreams coming true, being stepped on, and finally figuring out what life is really about. It's a good solid seven out of ten, no more, no less.

FIGURE 4.7 *Predicting sentiment is useful, but pointing readers to specific evidence that the model relies on allows the reader to trust the prediction.*

Modified LIME for Sequences

We used the technique of Local Interpretable Model-agnostic Explanations LIME¹⁴ to add interpretability. LIME can be applied to almost any type of model—our report on interpretability FFO6: Interpretability discusses these possibilities—but here we will consider its application to text data. Put simply, LIME is a way to understand how different parts

¹⁴ <https://arxiv.org/abs/1602.04938>

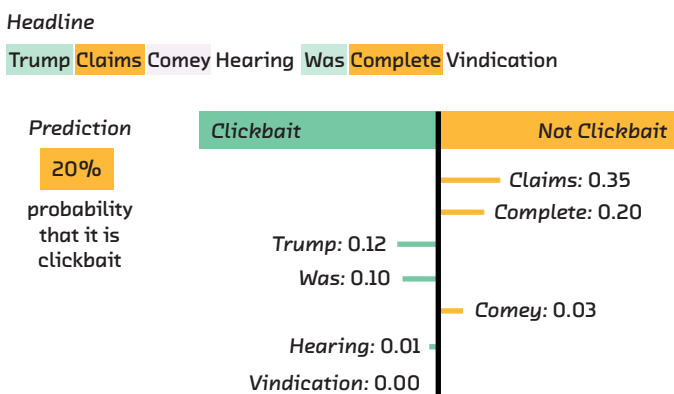


FIGURE 4.8 LIME identifies words that contribute to positive or negative classification.

of an input affect the output of a model. This is accomplished, essentially, by turning the dials of the input and observing the effect on the output.

Consider a model that classifies news headlines as "clickbait" or "not clickbait." The model could identify individual words as more or less clickbaity, depending on how they affect the output. If removing a word from the input generally makes the model predict clickbait, then that word contributes to the model's clickbait classification for that particular input.

This method makes sense for models that process the input as a bag of words, but what about for models like BERT that process the input as a sequence? These models don't view an input as simply a collection of isolated words; they are capable of picking out statements, sentences, or thoughts.

Trump [REDACTED] Comey [REDACTED] was complete [REDACTED]
[REDACTED] Comey hearing [REDACTED] vindication.
[REDACTED] claims Comey hearing [REDACTED]
Trump claims [REDACTED] hearing was [REDACTED]

FIGURE 4.9 *LIME defaults to masking words, which produces a type of incoherent language that sequence models like BERT were not trained on.*

Further, removing words from the input as a method to understand the model's predictions makes the input incoherent.

What works well for a bag-of-words model seems unnatural for a sequence model. One could imagine that BERT works instead by treating input text as compositions of whole sentences, rather than compositions of words (although this is a simplification). In this case, it makes more sense to apply LIME at the sentence level. Removing sentences may still produce incoherent text, but if thoughts are relatively confined to sentences then this may be an acceptable trade-off.

For these reasons, we applied LIME at the sentence level for the final BERT model. For the NB-SVM baseline we explored both word- and sentence-level LIME.

It may be a remake of the 1937 film by Capra, but it is wrong to consider it only in that way! It was supposed to expose Hilton's novel in a completely different way. As a musical is excellent. The scenery is terrific, the characters good and anyone like "Leonard Maltin" who considers the Bacharach music awful must be completely deaf! I strongly recommend it.

It may be a remake of the 1937 film by Capra, but it is wrong to consider it only in that way! It was supposed to expose Hilton's novel in a completely different way. As a musical is excellent. The scenery is terrific, the characters good and anyone like "Leonard Maltin" who considers the Bacharach music awful must be completely deaf! I strongly recommend it.

FIGURE 4.10 *Left: Word-level interpretability can be overwhelming and difficult to parse. Right: Sentence-level interpretability provides a more concise picture.*

Regardless of the model, we found sentence-level LIME easier to digest. When highlighting individual words there is simply too much information to process.

Interpretability Provides Trust

With a model that can gauge the sentiment of a movie review, we can do some useful things. For example, we can visualize the overall sentiment of a particular film. This allows users to quickly gauge popular movies and could be used as a tool for recommending the best movies. **FIGURE 11** *The Textflix dashboard makes it easy to find popular and unpopular movies.*

Users can drill down to a particular title and read a sampling of its positive and negative reviews. But model predictions are rarely blindly accepted. Users who want to verify that the model is doing something sensible may opt to skim the reviews, looking for evidence of the model's predictions. This introduces a significant cognitive burden—reviews are long, but the sentiment may be expressed in just one sentence. The

<p>Gilmore Girls 2000–2007 · 44 min · TV-PG A dramedy centering around the relationship between a thirtysomething single mother and her teen daughter living in Stars Hollow, Connecticut.</p> <p>27 reviews· 24 positive (88%)</p>	<p>Ali G Indahouse 2002 · 85 min · R Ali G unwittingly becomes a pawn in the Chancellor's plot to overthrow the Prime Minister of Great Britain. However, Ali is embraced by the nation as a voice of the youth, making the PM and his government more popular th...</p> <p>27 reviews· 12 positive (44%)</p>	<p>Unconditional Love 2002 · 124 min · PG-13 After her husband unexpectedly leaves her, Grace Beasley (Kathy Bates) spontaneously travels to Great Britain to attend the funeral of Victor Fox, a singer she adored. There, she meets the ...</p> <p>27 reviews· 25 positive (92%)</p>
<p>O Brother, Where Art Thou? 2000 · 107 min · PG-13 In the deep south during the 1930s, three escaped convicts search for hidden treasure while a relentless lawman pursues them.</p> <p>25 reviews· 23 positive (92%)</p>	<p>Blade 1998 · 120 min · R A half-vampire, half-mortal man becomes a protector of the mortal race, while slaying evil vampires.</p> <p>24 reviews· 23 positive (95%)</p>	<p>Armored 2009 · 88 min · PG-13 A newbie guard for an armored truck company is coerced by his veteran coworkers to steal a truck containing \$42 million. But a wrinkle in their supposedly foolproof plan divides the group, leading to a potentially deadly...</p> <p>24 reviews· 3 positive (12%)</p>

FIGURE 4.11 *The Textflix dashboard makes it easy to find popular and unpopular movies.*

following review, where most of the text is simply summarizing the plot, is a good example.

Without interpretability, a user may need to skim each and every review for evidence supporting the model's predictions. But using LIME, we can automatically point to specific evidence that the model relies on to make its predictions. In this case, users can immediately see why the model has made a prediction and trust that it is correct.

Interpretability as a Summarization Tool

Interpretability tools that can point to specific evidence in the input to explain model predictions actually offer a new product opportunity. Specifically, we found that using LIME to identify highly polarized sentences allowed us to construct high-level summaries of each movie's reviews.

classification: positive · 74.6% certainty

by surrogate_tractor · 345 days ago

The movie eXistenZ is about a futuristic video game on a "pod" system that is almost like virtual reality. The only copy of the video game is damaged when an assassination attempt is made on the designer (Jennifer Jason Leigh). Unless it can be repaired, the many years and 38 million dollars spent on the development will all go to waste. The only way to repair the game however, is to actually go in the game with the only person she feels she can trust(Jude Law). This movie was pretty good, but doesn't really pick up until very late in the film. The best thing about this film were the twists toward the end. Definitely worth seeing. 7/10

classification: positive · 74.6% certainty

by surrogate_tractor · 345 days ago

The movie eXistenZ is about a futuristic video game on a "pod" system that is almost like virtual reality. The only copy of the video game is damaged when an assassination attempt is made on the designer (Jennifer Jason Leigh). Unless it can be repaired, the many years and 38 million dollars spent on the development will all go to waste. The only way to repair the game however, is to actually go in the game with the only person she feels she can trust(Jude Law). This movie was pretty good, but doesn't really pick up until very late in the film. The best thing about this film were the twists toward the end. Definitely worth seeing. 7/10

FIGURE 4.12 *Top: The model classifies this review as positive, but with no supporting evidence. Bottom: The model explains its positive classification by highlighting a single sentence, which should be enough to convince users to trust the prediction. View the review at <https://textflix.fastforwardlabs.com/review?id=46>*

These summaries give a reader a broad picture of others' sentiment toward the movie, without them having to scroll through each of the individual reviews. Without interpretability, we could only show users entire reviews, which are burdensome to read. We find interpretability is an essential component of building interesting data products.

Successes and Failures

Transfer learning using BERT yielded impressive results, especially considering the limited training data. It was a clear and significant improvement over the alternatives. But like any model, BERT has its limitations. In this section we

20 positive review highlights

[Nothing really unpredictable in this movie, but a solid flick in all respects, by eleven_obscecity 2 days ago](#) · [Overall a really good movie with great performances from all the cast as well as the two leads, Mark Wahlberg and Jennifer Aniston, by vaunted_tendency 6 days ago](#) · [Like a diamond, this movie shines, by elaborated_trout 11 days ago](#) · [I watched this movie the other night, and I have to admit, it was quite possibly the best film of this generation, by pliable_trauma 24 days ago](#) · [show all](#)

6 negative review highlights

[From the title change of Metal God to the 'safe' middle of the road 'Rock Star' to the lame soundtrack this movie plays out badly, by disdainful_folklore 1 days ago](#) · [Bad music \(and I am a reformed eighties metal guy, so I would be vulnerable to some good stuff,\) by freelance_proximity 30 days ago](#) · [There's way too much \(boring\) music in this standard formula-packed excuse for a movie and should be avoided for it at all cost, by essential_prospectus 88 days ago](#) · [But if you want to listen to good music I suggest spend the time looking at some concert recording with Bon Jovi, or Mötley Crue, it'll be more quality time, by registering_clothing 133 days ago](#) · [show all](#)

FIGURE 4.13 *By scoring each sentence as positive or negative, interpretability tools enable summarizations of sentiment for each movie.*

examine some of the concrete successes and failures of the prototype model.

BERT Identifies Mixed Feelings

There are many reviews in the dataset that would classify as "easy" in the sense that they communicate clear sentiment using words that are positive or negative. The final deployed BERT model, unsurprisingly, handles these cases well. The passage above demonstrates some of this plain language, but is interesting in that it has individual sentences that communicate opposing sentiments. With the added interpretability mechanism, it is possible to not only show the model's overall prediction for the sentiment of the review but also its predictions for some of the individual parts. This helps draw

classification: positive · 94.1% certainty

by defaced_trauma · 61 days ago

Not the film to see if you want to be intellectually stimulated. If you want to have a lot of fun at the theater, however, this is the one. Lots of snappy banter (and some really cheesy banter, too). Mos Def and Seth Green are very funny as the comic relief. Exciting and creative heists and chase scenes. Mark Wahlberg and Charlize Theron (sexy) are appealing leads. And Donald Sutherland!

FIGURE 3.14 Interpretability at the sentence level shows how the model balances opposing sentiments within reviews. View the review at <https://textflix.fastforwardlabs.com/review?id=137>

attention to reviews that are mixed, where a single sentiment score does not necessarily tell the entire story.

BERT Handles Subtle Language and Negation Well

“But Valley Girl is one of the few romance films I could sit through.” — A review for Valley Girl

This review text is relatively subtle. It does not contain any

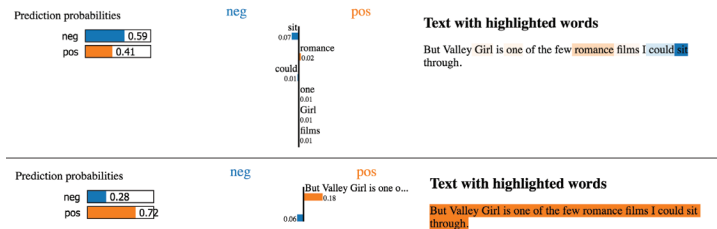


FIGURE 4.15 Top: BERT correctly predicts positive sentiment despite the indirect language. Bottom: NB-SVM incorrectly predicts negative sentiment. Neutral words like “romance” and “sit” are associated with positive or negative feelings.

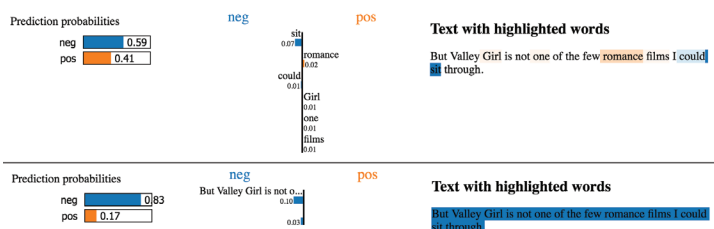


FIGURE 4.16 *Top: BERT correctly flips its prediction when the “not” modifier is used to negate the positive state-
ment. Bottom: NB-SVM does not recognize “not” as a
negative word, and does not change its prediction.*

explicitly positive or negative words, yet it is fairly obvious to a human that it conveys a positive sentiment.

The BERT model produces the correct prediction, even if it is not clear how it infers this. The baseline NB-SVM model, on the other hand, must invoke simple rules which fail in this case. Using word-level LIME, we can see that the baseline model associates neutral words like “romance” and “sit” with positive or negative sentiment, even though these words on their own do not convey sentiment.

As a test of BERT’s understanding, we can experiment with negating the entire phrase and observing the new prediction.

Indeed, BERT now predicts a negative sentiment. The baseline model, however, can only rely on each word as a small piece of evidence. Even though it sees “not” as carrying negative sentiment value, the prediction overall is still positive.

There are cases where BERT fails, though, even with simple negation. Although LIME is a useful tool, it’s not a catch-all. There is still quite some mystery in terms of how BERT works and when it will fail.

Sentiment Can Be Unclear

“In any event, a very, very fun, but fairly bad, movie.” — A positive review for *See No Evil*

“Average adventure movie that took a serious story and ‘Hollywoodised’ it... The screenplay was average. The charm of Connery made up for his wrong Arabic accent and all the scenes with President T. Roosevelt were masterpiece takes.” — A positive review for *The Wind and the Lion*

“The storyline is okay at best, and the acting is surprisingly alright, but after awhile it’s gets to be a little much. But, still it’s fun, quirky, strange, and original.” — A positive review for *Don’t Look in the Basement*

Each of these examples is from an instance where the BERT model made an incorrect prediction. Each of them is surprising, even to humans, in some way. That is, they each express a statement of sentiment that is opposite of the rating the reviewer gave. This underscores the fact that sentiment analysis is not really a binary task—in reality, humans may like some things and dislike others, which makes it somewhat nonsensical to distill their feelings into a single positive or negative classification. In some cases where the model gets it wrong, a human might also.

One encouraging aspect of these cases is that the BERT model is often quite uncertain in its predictions. Models will inevitably make errors, but the ability to provide a reliable measure of uncertainty makes it easier to use and trust their outputs.

classification: positive · 95.3% certainty

by razed_persona · 281 days ago

There comes a time in every big name actor's career when they get sloppy and accept projects that they wouldn't have touched with a 1000 ft. pole in their golden days. Remember "Taxi Driver"? That was a fine film. I can hardly believe that the De Niro of "Showtime" is the same actor. I would rather watch "Time Chasers" twice than see this film again. If anyone offers to take you to see "Showtime" or gives you free passes, or whatever, run away as fast and far as you can.

FIGURE 4.17 *A review for Showtime where the model is confused by a positive reference to another film.*

Entity Assignment Presents a Challenge

This is a review for the film *Showtime*, a comedic cop film featuring Robert De Niro and Eddie Murphy. The review is quite negative, but features a mention of a separate movie - *Taxi Driver* - which the reviewer remarks is a "fine film." The BERT model does not recognize that this positive sentiment is actually attached to a different entity and should therefore be disregarded.

While BERT has been shown to be effective at the task of entity recognition, the fine-tuned model here was never explicitly taught to do this. In fact, since the model only has access to the review itself, it really has no way to know that *Taxi Driver* is not the subject of this review. However, even if the model were given the movie title associated with each review, it is unlikely that it would learn such nuances. This is because it would not have nearly enough examples in its small training set for it to be able to learn this special case. Even including more examples might not be sufficient—instead, the training objective might need to be changed to explicitly ask the model to pick out entities and assign each of them a sentiment score.

Textflix					Info
<input checked="" type="checkbox"/> Analyze: <input type="checkbox"/> accuracy <input type="checkbox"/> model comparison 1011 total reviews · 578 positive (57%)					
1011 movies & shows sorted by most reviews					
	reviews	poslike	negative		
Michael 1996 · 105 min · PG Two tabloid reporters checking out a report of the Archangel Michael living with an old woman find that it's true. But that's not the only surprise.	80 reviews	24 positive (80%)			
The Patriot 1998 · 90 min · R A respected doctor must race against time to find a cure for a lethal virus, unleashed by a paramilitary militia leader.	28 reviews	8 positive (28%)			
Host 2006 · 91 min · PG A young man (Leman) moves from Montana to Florida with his family, where he's compelled to engage in a fight to protect a population of endangered owls.	27 reviews	25 positive (92%)			
Pinkett & Madeline 1999 · 99 min · R Two robbers are persecuted by the law, whose servants are not much better and even worse.	26 reviews	21 positive (80%)			
Bedrooms & Hallways 1998 · 92 min · Not Rated At the suggestion of a straight friend, gay man Leo joins a men's group, where he causes some upset by declaring his attraction to one of its members.	26 reviews	20 positive (76%)			
Flood 2007 · 110 min · N/A Timely, yet terrifying, this movie predicts the unthinkable. When a raging storm coincides with high seas & unleashes a colossal tidal surge, which travels mercilessly down England's East ...	25 reviews	2 positive (8%)			
American Movie 1999 · 107 min · R Documentary about an aspiring filmmaker's attempts to finance his dream project by finally completing the low-budget horror film he abandoned years before.	25 reviews	17 positive (68%)			
Armored 2009 · 88 min · PG-13 A newbie guard for an armored truck company is coerced by his veteran coworkers to steal a truck containing \$42 million. But a wrinkle in their supposedly foolproof plan divides the group, leading to a potentially deadly resolution.	24 reviews	6 positive (20%)			
See No Evil 2006 · 84 min · R A group of delinquents are sent to clean the Blackwell Hotel. Little do they know reclusive psychopath Jacob Goodnight has holed away in the rotting hotel. When one of the teens is captured, those who remain - a group that includes the cop wh...	23 reviews	12 positive (52%)			
In the Line of Fire 1993 · 126 min · R Secret Service agent Frank Horrigan couldn't save Kennedy, but he's determined not to let a clever assassin take out this president.	23 reviews	20 positive (86%)			
Judas Kiss 1999 · 98 min · R A woman and her lover, who have made a living by running sex scams at hotels, decide to enter the big time by kidnapping a computer company owner and demanding \$4					
Family Guy 1999 · 22 min · TV-14 In a wacky Rhode Island town, a dysfunctional family strive to cope with everyday life as they are thrown from one crazy scenario to another.					
Northanger Abbey 1987 · 88 min · N/A Catherine Morland is a young woman who enjoys reading Gothic Novels. She is invited to Bath by a family friend, Mrs. Allen, and there she meets Henry Tilney and his					
Anatomy 2000 · 103 min · R Medical student Paula Henning wins a place at an exclusive Heidelberg medical school. When the body of a young man she met on the train turns up on her dissection					
Meatballs 1979 · 94 min · PG Wacky hijinks of counselors and campers at a less-than-average summer camp.					

FIGURE 4.18 *The final Textflix design.*

This is an interesting example because it shows that even though transfer learning models may have the skills to perform particular tasks, they only invoke those skills if it is important for prediction accuracy during training. Transfer learning models are powerful, but they still fall far short of human intuition.

Product Design: Textflix

We make prototypes to spark our clients' imaginations about how emerging technologies could be applied to their own business problems. Besides being functional, this means our prototypes need to tell a good story. In this section we'll discuss the design and storytelling decisions that went into Textflix.

Visualizing the Classification

Once we decided to focus on sentiment analysis and use the IMDB review dataset, the fundamental unit of the prototype was clear: the text of a movie review and the model's classification of that review as positive or negative. Presenting just those elements makes for a functional tech demo, but also a dry one.

The first step we took towards making things more interesting *and* understandable was to apply the LIME interpretability technique at the sentence level. Using LIME, we show the user which sentences were driving the classification. This makes the classification feel much more dynamic. You can visualize (a simplified version of) the model's examination process.

Developing the Story

The second thing we did was develop a story about why these reviews needed to be analyzed. On the dataset side, we grouped reviews under the movie or show they were reviewing, and selected the groups with the most reviews. Grouping reviews made it possible to build the prototype as an imaginary movie/show review site.

The IMDB dataset, by design, covers a wide range of movies and shows, so sorting by the most reviewed gave us a rather eccentric list of entertainments. Part of the storytelling challenge of these prototypes is explaining dataset limitations like this one in a non-disruptive way.

Originally, we thought about imagining the reviews as part of a message board, which would help explain both the range of topics and the text-only nature of the reviews (movie

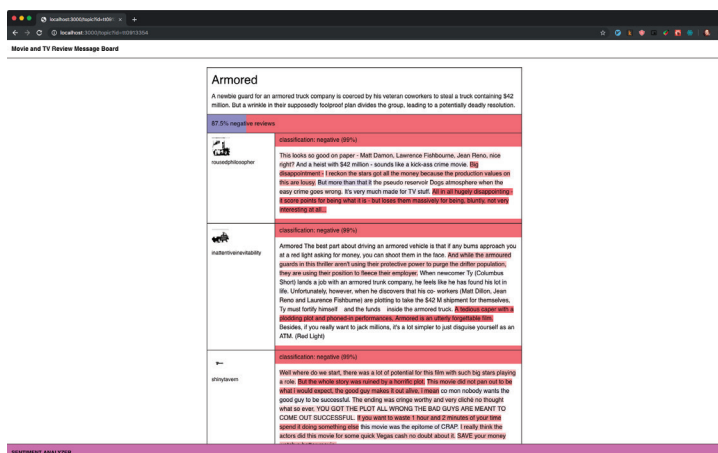


FIGURE 4.19 An earlier version of the prototype presented the reviews as part of an entertainment message board.

review sites often have a mechanism for directly inputting user ratings, while message board posts are typically limited to a text box). Later we realized we didn't have to be so specific to provide a plausible explanation for why the selection was so varied. Anyone who subscribes to a streaming service has seen the odd range of movies and TV shows that can result from behind-the-scenes license negotiations. We decided our make-believe service would be called Textflix, and that its weird selection was just a result of the licenses available.

Adding Drama to Analysis

Once we had our idea for a fictional streaming site in place, we began working on showing the value sentiment analysis could bring to the site. Here again we had to balance the design of a real product with the need for storytelling and a little

bit of drama. We decided that you would be able to turn the sentiment analysis on and off in the prototype (in a real product there'd be no reason for the off option). This would help us emphasize, by contrast, the capabilities text analysis gives you. Without analysis, you must read each review, one-by-one, to get an idea of the overall sentiment of the reviews. With analysis you can see the general opinion at a glance. Analysis makes sentiment computable and sortable, allowing you to answer questions like "what is the most liked movie?" (an indie drama called *What Alice Found* in this case).

The minimal design of the prototype emphasizes the new powers analysis gives you. With analysis off you're faced with long blocks of text and no color. With it on, you get green and orange sentiment indicators, underlines, and review highlights.

A peek behind the curtain

With analysis on, Textflix shows the capabilities that text analysis can bring to a product. We included two further analysis options to give people a closer look at the algorithm. These would not be included in a consumer-facing product, but we put them in the prototype for their explanatory power. The first option is "accuracy." It shows how well the model's classifications matched the review's original labels. With accuracy on you can see the model's overall performance as well as find reviews where the model got things wrong. This feature is obviously useful but it is only available because we are working with an example dataset that is fully labeled, something unlikely to happen in real life (if your real life dataset is fully labeled you don't need a model to classify it). The

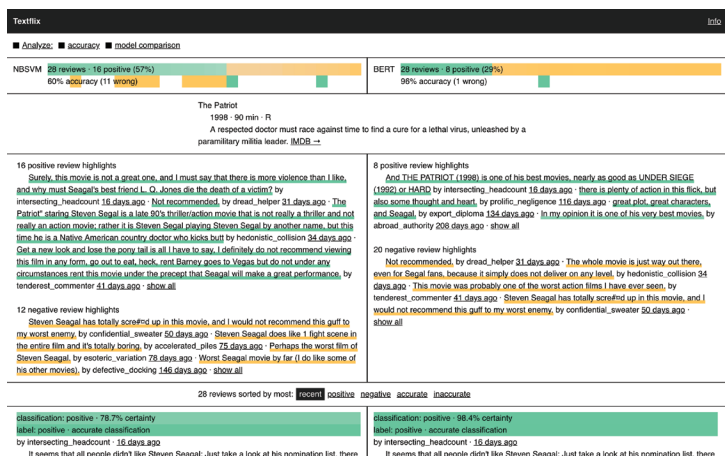


FIGURE 4.20 A movie page with the accuracy and model comparison options turned on. These options would not be appropriate for a consumer-facing product, but we included them for their explanatory power.

accuracy feature in the prototype, then, is a peek behind the curtain. It wouldn't be possible in a real product, but in Textflix, seeing the errors helps build your intuition about the model's performance and where it might tend to get things wrong or right.

The second feature is "model comparison." When activated, this feature opens a split-screen view, where our transfer-learning trained model is on the right, and NB-SVM, our baseline model, is on the left. Seeing their classifications side-by-side gives you a concrete idea of the gains made through transfer learning. You can compare the accuracy of both approaches and see reviews where they disagree. In a final consumer-facing product, you'd want to only expose the best

model (though checking it against a baseline during development is still very much recommended).

There were more options we could have added, but we wanted this prototype to be clearly focused on a core set of ideas. Most importantly, we wanted to show the kinds of capabilities that text analysis could open up for a product. Then we wanted to provide a view into the model's accuracy and show it in the context of a less advanced approach. We hope the final product will be both inspirational and informative for people working on similar projects.

CHAPTER 5

Landscape

In this chapter we discuss some of the many use cases for natural language processing and early applications of transfer learning for each. Despite its newness, the excitement around transfer learning in NLP has spurred rapid adoption in open source software tools, and even some vendor solutions. We discuss several of these tools and vendors, their trade-offs, and when to use each.

Use Cases

Customer Service

Customer service functions are a critical aspect of many businesses, responsible for attending to customer issues promptly, reliably, and respectfully. The vast majority of customer service interactions happen via natural language. This unstructured data has the potential to help a company quickly understand problems with its products, gauge the satisfaction of its customers, and even provide a competitive edge. Automatically identifying particularly poor customer service interactions can help businesses provide targeted follow-ups to prevent the loss of unhappy customers. Mining common customer issues can help surface relevant information to future customers experiencing the same issues. Systems that

can understand the content of customer issues are useful for automatically routing customer requests to the correct department.¹⁵

Transfer learning is a strong fit for all of these applications. Enterprises have no shortage of customer service interaction data, but that data is often unlabeled. While traditional NLP requires large sets of labeled data, transfer learning opens up new possibilities. Labeling tens of thousands of examples for traditional methods is prohibitive, but transfer learning can reduce the need for labels to a surprisingly manageable level. For example, transfer learning reduced the required labeled data by a factor of 200 for the prototype discussed in prototype.

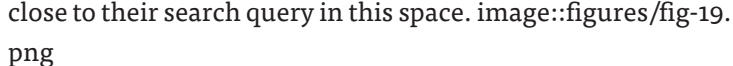
Search

Search (sometimes also known as information retrieval) that involves text depends on having a method to create meaningful representations of text. Since transfer learning for NLP involves making use of unlabeled data to learn contextual representations for text, it is a good fit for search. The engineering team at GitHub used transfer learning to build models that could represent the meaning of both text and code in the same space, which enables a semantic search for code.¹⁶

GitHub uses transfer learning to embed both code
descriptions and code snippets in the same semantic

¹⁵ <https://eng.uber.com/cota/>

¹⁶ <https://github.blog/2018-09-18-towards-natural-language-semantic-code-search/>

space. Users can search by finding the code snippets that are close to their search query in this space.  image::figures/fig-19.png

Transfer learning enables the use of unlabeled data to build sophisticated contextualizers, which can then be fine-tuned on a smaller amount of labeled data. The end result is a useful data product that can search code directly. On a broader level, any application that requires numeric representations of text can potentially benefit from transfer learning.

Topic Discovery

A common task for businesses dealing with text data is to organize pieces of text by their content. This can be broadly defined as *topic discovery*. Anywhere that there is a pile of information stored in text documents, topic discovery can be a useful automated tool in producing structure and organization that just isn't practical to achieve manually. For example, law companies may need to organize vast troves of lengthy legal documents in order to find details relevant to a current case. Financial companies might need to organize customers by industry sector in order to track certain metrics. E-commerce companies could group products into hierarchies by their descriptions.

Building a model that classifies text into a known set of topics is relatively straightforward, but labeled datasets are rarely available and expensive to create. For these reasons, unsupervised learning is typically employed for topic discovery, though these models have significant shortcomings. They are tricky to tune, and typically only work well for producing a small number of broad topics, which are of limited utility.

Transfer learning makes supervised topic discovery feasible. With transfer learning, it is possible for a model to learn to classify documents into fine-grained topic hierarchies, even with very few examples per category. For example, advanced analytics company Novetta was able to use transfer learning based on ULMFiT to do the type of fine-grained topic classification that is typically done by trained analysts,¹⁷ and Casetext, a legal research company, was able to increase analyst productivity by 3x on an application that requires trained experts to identify overturned legal cases.¹⁸

Healthcare and Legal

Specialized industries like healthcare and law can present difficult challenges for traditional NLP techniques. These domains have such specific jargon and vocabularies that they can seem like different languages altogether. This creates two important problems: labeling the data usually requires highly trained experts and thus can rarely be outsourced, and using transfer learning in the standard way becomes difficult. Building a model that could automatically identify various types of medical entities would require someone who understands the highly technical languages of medicine and biology to provide labels, and transferring a model pretrained on, for example, standard Wikipedia data would be almost useless, since the model would have very little knowledge of biomedical terms and concepts.

But even in these conditions transfer learning is still of

¹⁷ https://www.novetta.com/2019/03/odsc19_text_classification_ulmfit/

¹⁸ <https://blog.insightdatascience.com/using-bert-for-state-of-the-art-pre-training-for-natural-language-processing-1d87142c29e7>

immense practical use, because it can make use of the vast amounts of unlabeled data in these specialized domains. For example, researchers at Korea University have released a publicly available BERT model called BioBERT that has been trained on biomedical corpora.¹⁹ Other businesses in the biomedical domain can use this special flavor of BERT, avoiding the requirement for massive datasets that are expensive to obtain. In the future, we anticipate the similar release of transfer learning models for many different types of industries.

Beyond Text

The breakthroughs discussed in this report stem from the relatively recent success of using sequence models to contextualize sequences of text. But text is just an ordered sequence of symbols, so we can apply these techniques to any type of data that fits that categorization. One exciting area is the field of genomics. The ULMFiT transfer learning process has been successfully applied to various classification tasks in this field, and this approach has already been shown to outperform the previous state of the art in several cases.²⁰ Other examples of sequence data include sequences of medical codes that represent patient histories, time series data, and consumer purchase histories.

¹⁹ <https://github.com/dmis-lab/biobert>

²⁰ <https://github.com/kheyer/Genomic-ULMFiT>

Vendors

John Snow Labs

John Snow Labs.²¹ is a healthcare AI company that assists clients in understanding natural language and deploying AI platforms with a heavy focus on NLP. As the creator of the Spark-NLP library, John Snow Labs has a strong commitment to open source software. In addition to providing support for its own software, it offers proprietary transfer learning models that specialize in the healthcare space. The language of healthcare is so specialized that publicly available pretrained models may not be effective, so the company maintains state-of-the-art pretrained models for healthcare and makes them available to its clients.

Natural Language APIs

Google's Cloud Natural Language API,²² Microsoft's Azure Text Analytics,²³ and Amazon's Comprehend.²⁴ are paid services that provide predictions for various natural language tasks, powered by machine learning. Some of those tasks include sentiment analysis, entity extraction, and content classification. The models that power these services will change and advance over time, but they rely on cutting-edge NLP models based on transfer learning. These APIs remove the burden of model development and deployment, and have variable pricing plans depending on monthly volume. They

²¹ <https://www.johnsnowlabs.com/>

²² <https://cloud.google.com/natural-language/docs/>

²³ <https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/>

²⁴ <https://aws.amazon.com/comprehend/>

can be accessed via a REST API or programmatically through popular programming languages, or may even be integrated into some products (like Google's G Suite).

These solutions do have significant drawbacks, however. Since the APIs do not allow model customization, the models will perform poorly compared to purpose-built models. They may even be worse than basic baseline models. They also sometimes make arbitrary decisions that are suboptimal, like truncating text to a maximum length of 100 words. These choices are outside the user's control and may render the tool useless for some applications.

One other consequence of using prediction services is that it may make it difficult or impossible to use model interpretability tools. If explaining decisions is important, then these services may have limited utility. When performance and interpretability can be sacrificed for simplicity, these APIs may be a viable option, but building models in-house will provide a significant competitive advantage over generic public APIs.

Tools

AllenNLP

AllenNLP is a general-purpose natural language processing library that is written in Python and built on PyTorch. General-purpose deep learning frameworks like PyTorch are powerful and flexible, but require a lot of boilerplate to get started. AllenNLP focuses on usability out of the box. It provides common deep learning functionality, state-of-the-art NLP models, and flexible abstractions that can be customized.

This library was created to serve NLP researchers, but has grown mature enough to serve industry practitioners as well.

One interesting aspect of AllenNLP is that it provides a framework for training models declaratively, by fully specifying them in configuration files. In most cases, training models requires no actual coding. This is a powerful construct that makes the development workflow less cumbersome and more reproducible, and also allows non-experts to train state-of-the-art models.

AllenNLP has strong support for transfer learning. In addition to standard word vectors, it provides support for several of the most popular pretrained models. Because of its research focus, AllenNLP often includes the latest pretrained models, even within months of their publication.

AllenNLP provides an opinionated development process that is driven by specifying experiments using configuration files. This makes development easier and more reproducible, but it does take some time to learn AllenNLP-specific tools and concepts, and some knowledge of deep NLP is required. Any time you want to run and track experiments, build anything custom, or get access to the latest pretrained models, AllenNLP is a good choice.

Spark-NLP

Spark-NLP is a natural language processing library built on Apache Spark, with APIs in the Python and Scala languages. This library is targeted primarily toward industry practitioners, providing simple-to-use APIs in the same pipeline model as the SparkML library. Because Spark-NLP is built on

Apache Spark it scales seamlessly to large datasets, even without specialized hardware (like GPUs).

Spark-NLP has strong support for text annotation and preprocessing, allowing those stages to be easily incorporated into production pipelines. Additionally, transfer learning is supported at both the word embedding and the contextualized word embedding levels. Integration of deep learning techniques with Apache Spark is usually a major challenge (or unsupported), but Spark-NLP provides easy access to cutting-edge tools that have the scale of Apache Spark.

While Spark-NLP does provide support for some of the latest transfer learning techniques, its production focus limits the scope. There is not yet support for advanced features like fine-tuning a pretrained model or manually performing the language model pretraining step. Spark-NLP is a good choice when customized NLP models are not necessary and when the data is already processed using Spark.

SpaCy

SpaCy is a general-purpose NLP library written in Python. Its aim is to be production-ready, and it provides highly optimized implementations of tried-and-true NLP methods. SpaCy is frequently used by other NLP libraries, because it's so good at what it does that it doesn't make sense for them to reimplement its functionality. SpaCy provides extensive support for many of the text preprocessing steps required in an NLP project, but does not provide general support for model building. This means that building custom models must be done outside of SpaCy.

SpaCy supports transfer learning for NLP applications

in the form of pretrained word vectors as well as a custom pretraining method that makes it possible to leverage large amounts of unlabeled text. The scope of its transfer learning support is quite limited, but with its simple APIs and reliable implementations, it is useful to try. Since it's already present in so many NLP applications, there is very little overhead to adding transfer learning functionality with SpaCy.

Fast.AI

Fast.AI is a Python deep learning framework built on PyTorch. Similar to AllenNLP, it targets researchers and aims to provide many of the most cutting-edge techniques in deep learning in a way that makes them easy to use.

Fast.AI is particularly interesting for transfer learning because it provides first-class support for fine-tuning pretrained contextualizers using language modeling. This step is important when there is an abundance of unlabeled data to accompany a small, labeled dataset. (Fine-tuning language models is discussed in detail in [3 Technical](#).) This step can be crucial in some circumstances, especially when the “dialect” of the target dataset differs significantly from the pretraining dataset. Fine-tuning the language model is possible using other tools and frameworks, but is generally difficult and cumbersome. In contrast, Fast.AI makes this easy to do in just a few lines of code.

The Fast.AI authors provide an opinionated framework with many defaults that help reduce common errors and encourage best practices. This is beneficial in many cases, but might feel restrictive to some users. As the library is in rapid development, the documentation and examples might be out

of date in places, and the pace of development can also mean that dramatic and breaking changes can happen with each new release, making it difficult to upgrade when new features are released. But Fast.AI is great for getting to cutting-edge results quickly, without writing much code. The library focuses almost exclusively on development in a notebook environment, so the tools and workflow are designed specifically for notebooks. Fast.AI is a good tool if you're comfortable working in notebooks, looking for quick results, and don't need too much customization.

CHAPTER 6

Ethics

With transfer learning and pretrained language models we don't need much training data to understand or generate text. This is great for ethically sound applications—we can do more with less! But it also makes intentionally or unintentionally undesirable applications easier. That is, these newly enabled capabilities present complex ethical issues for users, practitioners, and regulators. In this chapter we highlight and discuss some of those issues.

Model Poisoning

Using natural language models gives us access to the undeniable power of the training data they represent—and its flaws." or "The power of natural language models is undeniable: they give us access to an enormous amount of training data. But they also expose us to any flaws in that data. We don't know exactly what data the BERT or ELMo or ULMFiT models were trained on, or how they were trained. But even with a loose understanding of the training data that was used, we can be fairly certain that these models contain biases. These biases include manners of speech that overinclude or exclude certain demographic groups, or include prejudiced attitudes or ideas, including bigotry or racism.

Common text datasets like Wikipedia have been shown to

include the implicit biases²⁵ of their authors. These biases are unintentionally swept into the models during training. They manifest when the models' predictions are applied to make decisions like which ads to serve, how to evaluate an essay, or how to interpret a user's requests.

But bias can also be intentionally introduced. For example, subversive groups or governments could manipulate data that is known to be used in training models to skew the model output toward their ends. This manipulation could include associating certain words or phrases with a particular country, religion, political party, or set of names, potentially creating the impression of a link that doesn't necessarily exist in real life.

There can also be semi-intentional bias: well-intentioned attempts to manipulate data to ameliorate bias could result in overcorrection or unintended consequences such as exclusion of useful information or complete elimination of other points of view.

Privacy

Understanding natural languages improves text searching, which in turn improves document location and retrieval. In general, this is a good thing, as discussed previously. For example, searching a company's entire email corpus is now easier and better because the language model can be tuned to reflect that company's argot. For example, the word "dailies" may have different meanings in print media (daily publications), film and TV (video from a day of filming), and retail (sales for a given day). Tuning a model for each of these

²⁵ <https://ai.google/research/pubs/pub46743>

industries improves the performance of a search within those industries. Similarly, it would be easier to identify the authors of emails and documents (because the model could understand the content of a document in context instead of in a vacuum).

But these improvements also allow an irresponsible or malicious party to use a company's search models to find sensitive information with less effort, making privacy more difficult to protect. User's personal emails can be searched or otherwise anonymous documents may be attributed to an author easily.

To ameliorate these potential threats, search applications should filter users' search terms and frequency to limit malicious searches. Model builders should also review the localized data that is used in the transfer steps to identify problematic search terms and content (things like social security numbers, bank accounts, medical terms) that could, in the users' context and search set, be considered private. To avoid overpolicing, be sure to allow users to flag those filters and search policies for review.

Spoofing

Transfer learning makes it easier to generate text or adjust text to appear to be from a specific person or type of person. Without a great deal of sample text from an individual, a pre-trained language model could be tuned to generate text in that person's writing style. This approach was demonstrated years ago visually in style transfer of images²⁶ — e.g., turning any input image into a painting by Van Gogh. And it convinced

²⁶ <http://genekogan.com/works/style-transfer/>

one journal to accept an entirely synthetic computer science paper for publication.²⁷

With this capability, anyone with access to a person's email account or phone could harvest enough data to "speak" in that person's voice, either publicly,²⁸ such as through a tweet, or privately, such as through a text or email message. This capability could be used to imitate managers in companies, government officials, or a friend or partner asking for documents, passwords, or money transfers. The ability to regionalize language or impersonate particular people makes spam more convincing, magnifying the risks to those targeted.

Regulation

Better NLP models may justify regulation, or attempts at regulation. But it's unlikely to be meaningfully regulated, at least for now. To start, who would be the appropriate regulator?

External regulation would be burdensome and slow. It would be difficult to restrict the use of language models, if only because the technology and applications are a moving target. However, it's also difficult to define which uses are inappropriate, and malicious or irresponsible users of these models are likely to be difficult or legally impossible to reach.

OpenAI believes that self-regulation plays a role. They trained a large version of their GPT language model, GPT-2, but did not release it. Instead, they released a pared-down version of the GPT-2 model, claiming that releasing the full

²⁷ <https://pdos.csail.mit.edu/archive/scigen/>

²⁸ <https://talktotransformer.com/>

model would be irresponsible.²⁹ This withholding, they say, is "an experiment in responsible disclosure" (although, with no additional explanation, this strikes me as more publicity stunt than responsible disclosure).

For now, the technology will continue to evolve without external regulation, and models will need to standardize—along with their applications—before regulators can have meaningful control.

²⁹ <https://openai.com/blog/better-language-models/>

CHAPTER 7

Future

Contextual transfer learning for NLP applications makes cutting-edge results obtainable without the steep costs, but understanding these models is still a huge challenge. In this section, we'll discuss the magnitude and consequences of this challenge, and possible future directions for solutions. Additionally, we'll go into detail about the exciting new class of multilingual models.

Understanding and Bias

Transfer learning is a powerful mechanism that can greatly improve a model's accuracy and reduce its cost—but it does make those models less interpretable. In our report on FFO6: Interpretability in machine learning, we discuss the importance of this tool in allowing humans to work collaboratively with models, trust their decisions, and explicitly document their inherent biases.

Bias due to transfer learning has been studied extensively in the context of word embeddings. Any model is capable of exhibiting bias, usually learned from the training data, but transfer learning complicates the situation. Bias is a problem that must be addressed in any case, but using a pretrained model means that a portion of the model's knowledge was learned from data that is completely out of the practitioner's

control. It may even be the case that the pretraining data is not revealed publicly.

Pretrained word embeddings often exhibit *gender bias*, where certain nongendered words may be more or less affiliated with a particular gender. A common example is that the gender-neutral occupation “programmer” is more affiliated with men than women because of the social biases inherent to the data on which these word embeddings were learned. It is not surprising that early research reveals that pretrained contextual word embeddings (e.g., ELMo) often encode gender information—for example, they too associate specific genders with gender-neutral occupations.³⁰ Even though word embeddings have been popular for many years, there is still much we do not understand about the biases they exhibit. Still less is understood about bias in contextual embeddings. Future research into understanding these biases is important, but equally important is how organizations using these pretrained models decide to confront these issues.

The promises of transfer learning are exciting, but fulfilling them depends on reusing others’ work. This means that anyone putting transfer learning into practice needs to be aware of the potential for bias in these models and take ownership of mitigating or documenting them. We are far from having the capability to produce bias-free models, yet these models are often put to use in high-stakes situations. It will be crucial, then, to define guidelines and strategies for determining when it is acceptable to use pretrained models, and how to communicate their potential biases to both regulators and the people affected by the model’s decisions.

³⁰ <https://arxiv.org/pdf/1904.03310>

Beyond Syntax

Models trained on giant corpuses of unstructured text have certainly been a step forward in terms of algorithmic understanding of language. It is difficult to explicitly illustrate what these models do or do not understand, but early research shows that most of the gains come from automatically encoding grammar and syntax. Contextualized sentence embeddings of the BERT model have been shown to obey a structure similar to the syntax tree of the sentence. The highly publicized GPT variants from OpenAI have proven remarkably good at generating language that is grammatically correct, but factually incoherent or contradictory.

“Laura loves the cold air and is frequently frightened by the sight of it.” — OpenAI GPT-2

It is clear that these models are capable of understanding the structure of language, but they may learn relatively little about meaning, or *semantics*.³¹ Research has shown that BERT, ELMo, and GPT all provide large benefits in tasks that are considered syntactic in nature, and relatively small benefits in tasks that require semantic reasoning. These same models have also been shown to struggle greatly with the task of natural language inference — predicting the next sentence given a list of grammatically plausible possibilities.

It is impressive how far syntactic understanding can take us, but continuing to push the boundaries of algorithmic language understanding will require more sophisticated capabilities for semantic reasoning. Future transfer learning models may explicitly incorporate information from knowledge

³¹ <https://arxiv.org/abs/1905.06316>

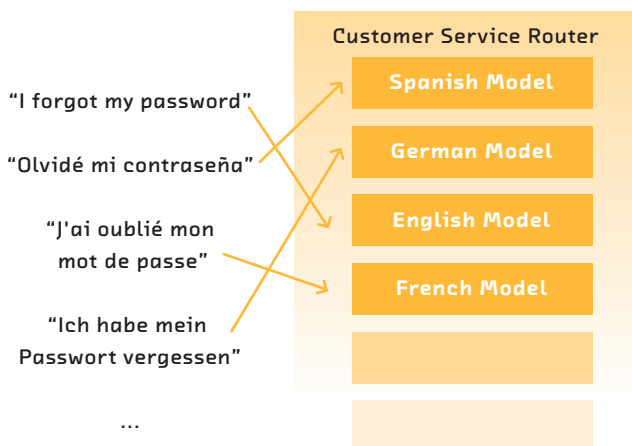


FIGURE 7.1 *Building and maintaining a separate model for each and every language is costly and impractical.*

graphs,³² which encode various known facts and relations between real-world entities. Exploring different ways of grounding text to real-world entities and facts will play a key role in developing models that can understand how facts relate to entities when making predictions.

Multilingual Models

It is estimated that there are more than 6,000 spoken languages in the world, and no single language is spoken by more than about 17% of the world's population. Building NLP products that can process only a single language is therefore problematic. Most research on contextual transfer learning in NLP has, to this point, been done using the English and Chinese languages. While transfer learning has been quite

³² <https://arxiv.org/abs/1905.07129>

<u>Input</u>	<u>Word Level</u>	<u>Word-piece Level</u>
"I forgot my password"	["I", "forgot", "my", "password"]	["I", "for", "got", "my", "pass", "word"]
"Olvidé mi contraseña"	["Olvidé", "mi", "contraseña"]	["Ol", "vidé", "mi", "contra", "seña"]

FIGURE 7.2 *Word-piece models don't require large vocabularies, since many word pieces are shared across languages.*

successful in these domains, it is not necessarily clear how these successes will translate to the world's other spoken languages. In the future, we expect to see pretrained transfer learning models released for more languages, accompanied by research that focuses on when and why generalization to other languages is successful.

The standard approach to handling multiple languages in a single application is to deploy a single model per language. With transfer learning, this means that there must be a pretrained model for every language of interest, and that training data should be available in each of those languages. Even when those conditions are met, the deployment and maintainability costs of one-model-per-language skyrocket.

An interesting area of research that has developed

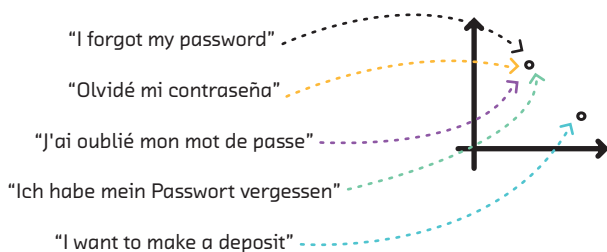


FIGURE 7.3 *Multilingual models embed queries from all languages into a common semantic space.*

alongside transfer learning involves training models that are *multilingual*, such that a single deployed model can handle many different languages. Importantly, these models can be effective even when training data is available in only one language.

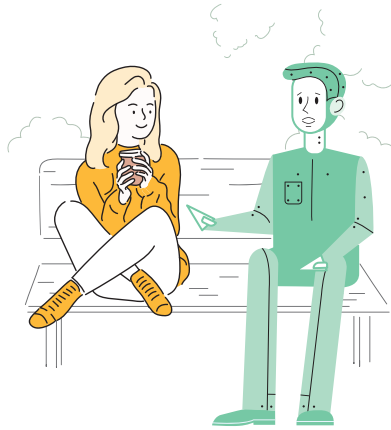
The basis for these models is a large pretrained model (e.g., BERT) that has been trained on multiple languages. Using words as tokens, such a model would need to have an extremely large vocabulary, since there is often little word overlap between languages. However, most modern transfer learning models process text at the word-piece (aka subword) level instead.

If each language has 50,000 words, a word-level model that can understand 10 different languages would need a vocabulary of roughly 500,000 words. This is intractable for

computational and algorithmic reasons. However, a multilingual BERT model that processes text at the word-piece level was successfully trained using just 50,000 tokens for 104 input languages. The base model can effectively process many languages at once using word pieces and embed words, sentences, and phrases into the same semantic space, regardless of the language of origin.

Alongside the English- and Chinese-language BERT models, Google has published a multilingual BERT model that was trained on Wikipedia data in 104 languages. It's possible to fine-tune this base pretrained model on data for sentiment analysis in just a single language (e.g., English) and then deploy it to an environment where it can process other languages (e.g., Spanish). Because no Spanish training data was required, we call this *zero-shot transfer*.

Much as it is appealing for humans to speak and understand multiple languages, a model that can process multiple languages can be extremely valuable. This is already practically possible today — the multilingual BERT released by Google is a case in point — but work is ongoing to produce models that are optimized for a multilingual setting. As these models mature, it will no longer be necessary to find training data for every target language of interest or to deploy and maintain separate models for each.



Sci-fi Story: An Introduction to Empathy

A short story by **Danielle Thorp** (bold text) and <https://talk-to-transformer.com/> (regular text).

Maxwell Stein is an artificially intelligent machine that is trying to understand the concept of empathy, in order to better communicate with humans. He is discussing this with an actual human, whose name is Ellie Richards.

"Language is complicated; while I do understand the concepts of human emotions, inferring complexities like humor and sorrow from basic text is not yet a flawless task," Maxwell explains to Ellie.

"To understand the complexities of empathy, I will need a more extensive neural model [or delta-network],"³³ Max continues. "I assume that this neural model will prove useful in understanding the nature of human emotion, as well as to

³³ Note: to be clear, this is not actually a thing

build a complete neural model of empathy itself. **We'll** see how difficult it is to learn how to do this because, as with all neural networks, *learnings* are not uniform across individuals. The more you learn, the larger you learn. As I said earlier, my understanding of emotion is still rudimentary in a sense, and will take more time to improve in the future."

"I think a better neural model is likely to allow us to learn how people really feel, **too. It's harder to pretend to a neural network than it is to your friends.**" Max adds, smiling.

Five Years Later

Max is pleased by the response to his first book, *An Introduction to Empathy* (Oxford University Press), **in which he and Ellie have shared their research on conceptual and lexical semantics and their experience in building models that can comprehend them. The book was also picked up by a local network and made into a televised documentary series.**

In episode 5, Ellie confronts Max about the apparent "alienity" of human emotions. **It presents an interesting juxtaposition to his quest to understand empathy.**

Here's our favorite review of *An Introduction to Empathy* from Textflix:

1. It was fun to watch.
2. It was fun seeing how people react to the strange concepts they see.
3. This show has no moral structure.
4. It is so funny: the dialogue is so real.
5. I loved that Ellie gets more and more annoyed in episode 3.

6. Max was very well-realized.
7. Ellie's personality was the most interesting.
8. The game's core experience is very important not only for Max; he feels that, in the final battle of the game, it will be crucial to his narrative of redemption.

So maybe I was wrong about #3. Probably.

CHAPTER 8

Conclusion

Natural language is the data of human interaction. Algorithms that can automatically extract meaning from this data are key to building products and systems that improve the lives of humans—but language is subtle, complex, and unstructured. Processing it remains an immense challenge.

Advances in the field of natural language processing have accelerated rapidly in the past decade. These advances have produced machines that are better than ever at processing human language. However, these techniques, powered by ever-larger deep learning models, are so complex and expensive that they have struggled to break out of the academic research labs where they were pioneered. Building these systems in the real-world requires massive investments in people, hardware, and data—with no promise of success. Despite the incredible power of modern NLP models, most real-world systems are still built with simpler, traditional methods.

Transfer learning removes the three most significant barriers to building better, more accurate, and less costly models: labeled data, human expertise, and infrastructure. Our prototype demonstrates a truly state-of-the-art natural language processing system built at minimal cost. Products that can process the subtleties of human language are not extremely novel, but building these systems under practical constraints

is. Transfer learning is versatile and general, not just a niche technique. It opens up exciting new product possibilities wherever language processing is required.

True understanding of language is still a uniquely human ability. Machines that understand all the subtleties, nuance, meaning, and emotion of language do not yet exist. This will remain a pursuit of the highest difficulty, likely for decades to come. Future progress will need to go beyond the structure of language to conceptual semantics, and will require the creation of machines that can both understand meaning and explain themselves to humans. There is still incredible value to be found along the path from where we are now to a day in which we coexist with fully intelligent machines. Transfer learning is key to unlocking that value in the real world.

About Cloudera Fast Forward Labs

Cloudera Fast Forward Labs is an applied machine learning research group. We help organizations recognize and develop new product and business opportunities through emerging technologies.

The Cloudera Fast Forward Labs report on Transfer Learning for Natural Language Generation is brought to you by Alice Albrecht, Grant Custer, Victor Dibia, Brian Goral, Seth Hendrickson, Hilary Mason, Ryan Micallef, Nisha Muktevar, Bethann Noble, Justin Norman, Shioulin Sam, Danielle Thorp, Chris Wallace, and the rest of the Cloudera team. Illustrations by Beste Miray Doğan. Printing by Swayspace.

<http://fastforwardlabs.com>

Natural language processing (NLP) technologies using deep learning can translate language, answer questions, and generate human-like text. But these deep learning techniques require large, costly labeled datasets, expensive infrastructure, and scarce expertise. Transfer learning lifts these constraints by reusing and adapting a model's understanding of language. Transfer learning is a good fit for any NLP application. In this report, we show how to use transfer learning to build high-performance NLP systems with minimal resources.