

1 Two Sea Changes in NLP


Fully supervised learning, where a task-specific model is trained solely on a dataset of input-output examples for the target task, has long played a central role in many machine learning tasks (Kotsiantis et al., 2007), and natural language processing (NLP) was no exception. Because such fully supervised datasets are ever-insufficient for learning high-quality models, early NLP models relied heavily on *feature engineering* (Tab. 1 a.; e.g. Lafferty et al. (2001); Guyon et al. (2002); Och et al. (2004); Zhang and Nivre (2011)), where NLP researchers or engineers used their domain knowledge to define and extract salient features from raw data and provide models with the appropriate inductive bias to learn from this limited data. With the advent of neural network models for NLP, salient features were learned jointly with the training of the model itself (Collobert et al., 2011; Bengio et al., 2013), and hence focus shifted to *architecture engineering*, where inductive bias was rather provided through the design of a suitable network architecture conducive to learning such features (Tab. 1 b.; e.g. Hochreiter and Schmidhuber (1997); Kalchbrenner et al. (2014); Chung et al. (2014); Kim (2014); Bahdanau et al. (2014); Vaswani et al. (2017)).¹

However, from 2017-2019 there was a sea change in the learning of NLP models, and this fully supervised paradigm is now playing an ever-shrinking role. Specifically, the standard shifted to the *pre-train and fine-tune* paradigm (Tab. 1 c.; e.g. Radford and Narasimhan (2018); Peters et al. (2018); Dong et al. (2019); Yang et al. (2019); Lewis et al. (2020a)). In this paradigm, a model with a fixed² architecture is *pre-trained* as a language model (LM), predicting the probability of observed textual data. Because the raw textual data necessary to train LMs is available in abundance, these LMs can be trained on large datasets, in the process learning robust general-purpose features of the language it is modeling. The above pre-trained LM will be then adapted to different downstream tasks by introducing additional parameters and *fine-tuning* them using task-specific objective functions. Within this paradigm, the focus turned mainly to *objective engineering*, designing the training objectives used at both the pre-training and fine-tuning stages. For example, Zhang et al. (2020a) show that introducing a loss function of predicting salient sentences from a document will lead to a better pre-trained model for text summarization. Notably, the main body of the pre-trained LM is generally (but not always; Peters et al. (2019)) fine-tuned as well to make it more suitable for solving the downstream task.

Now, as of this writing in 2021, we are in the middle of a second sea change, in which the “pre-train, fine-tune” procedure is replaced by one in which we dub “*pre-train, prompt, and predict*”. In this paradigm, instead of adapting pre-trained LMs to downstream tasks via objective engineering, downstream tasks are reformulated to look more like those solved during the original LM training with the help of a textual *prompt*. For example, when recognizing the emotion of a social media post, “I missed the bus today.”, we may continue with a prompt “I felt so ___”, and ask the LM to fill the blank with an emotion-bearing word. Or if we choose the prompt “English: I missed the bus today. French: ___”, an LM may be able to fill in the blank with a French translation. In this way, by selecting the appropriate prompts we can manipulate the model behavior so that the pre-trained LM itself can be used to *predict* the desired output, sometimes even without any additional task-specific training (Tab. 1 d.; e.g. Radford et al. (2019); Petroni et al. (2019); Brown et al. (2020); Raffel et al. (2020); Schick and Schütze (2021b); Gao et al. (2021)). The advantage of this method is that, given a suite of appropriate prompts, a single LM trained in an entirely unsupervised fashion can be used to solve a great number of tasks (Brown et al., 2020; Sun et al., 2021). However, as with most conceptually enticing prospects, there is a catch – this method introduces the necessity for *prompt engineering*, finding the most appropriate prompt to allow a LM to solve the task at hand.

This survey attempts to organize the current state of knowledge in this rapidly developing field by providing an overview and formal definition of prompting methods (§2), and an overview of the pre-trained language models that use these prompts (§3). This is followed by in-depth discussion of prompting methods, from basics such as prompt engineering (§4) and answer engineering (§5) to more advanced concepts such as multi-prompt learning methods (§6) and prompt-aware training methods (§7). We then organize the various applications to which prompt-based learning methods have been applied, and discuss how they interact with the choice of prompting method (§8). Finally, we attempt to situate the current state of prompting methods in the research ecosystem, making connections to other research fields (§9), suggesting some current challenging problems that may be ripe for further research (§10), and performing a meta-analysis of current research trends (§11).

Finally, in order to help beginners who are interested in this field learn more effectively, we highlight some systematic resources about prompt learning (as well as pre-training) provided both within this survey and on companion websites:

-  : A [website](#) of prompt-based learning that contains: frequent updates to this survey, related slides, etc.
- Fig. 1: A typology of important concepts for prompt-based learning.

¹Even during this stage, there was some use of pre-trained models exemplified by word2vec (Mikolov et al., 2013b,a) and GloVe (Pennington et al., 2014), but they were used for only a limited portion of the final model parameters.

²This paradigm is less conducive to architectural exploration because (i) unsupervised pre-training allows models to learn with fewer structural priors, and (ii) as pre-training of models is time-consuming, experimenting with structural variants is costly.

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	<div style="display: flex; justify-content: space-around;"> <div>CLS </div> <div>LM </div> <div>TAG </div> </div> <div style="text-align: center; margin-top: 10px;"> GEN </div>
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	<div style="display: flex; justify-content: space-around;"> <div>CLS </div> <div>LM </div> <div>TAG </div> </div> <div style="text-align: center; margin-top: 10px;"> GEN </div>
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Table 1: Four paradigms in NLP. The “**engineering**” column represents the type of engineering to be done to build strong systems. The “**task relation**” column, shows the relationship between language models (LM) and other NLP tasks (CLS: classification, TAG: sequence tagging, GEN: text generation). : fully unsupervised training. : fully supervised training. : Supervised training combined with unsupervised training. indicates a textual prompt. Dashed lines suggest that different tasks can be connected by sharing parameters of pre-trained models. “LM→Task” represents *adapting LMs (objectives) to downstream tasks* while “Task→LM” denotes *adapting downstream tasks (formulations) to LMs*.

- Tab.7: A systematic and comprehensive comparison among different prompting methods.
- Tab.10: An organization of commonly-used prompts.
- Tab.12: A timeline of prompt-based research works.
- Tab.13: A systematic and comprehensive comparison among different pre-trained LMs.

2 A Formal Description of Prompting

2.1 Supervised Learning in NLP

In a traditional supervised learning system for NLP, we take an **input** x , usually text, and predict an **output** y based on a model $P(y|x; \theta)$. y could be a label, text, or other variety of output. In order to learn the parameters θ of this model, we use a dataset containing pairs of inputs and outputs, and train a model to predict this conditional probability. We will illustrate this with two stereotypical examples.

First, *text classification* takes an input text x and predicts a label y from a fixed label set \mathcal{Y} . To give an example, sentiment analysis (Pang et al., 2002; Socher et al., 2013) may take an input x = “I love this movie.” and predict a label y = ++, out of a label set $\mathcal{Y} = \{++, +, \sim, -, --\}$.

Second, *conditional text generation* takes an input x and generates another text y . One example is machine translation (Koehn, 2009), where the input is text in one language such as the Finnish x = “Hyvää huomenta.” and the output is the English y = “Good morning”..

2.2 Prompting Basics

The main issue with supervised learning is that in order to train a model $P(y|x; \theta)$, it is necessary to have supervised data for the task, which for many tasks cannot be found in large amounts. Prompt-based learning methods for NLP attempt to circumvent this issue by instead learning an LM that models the probability $P(x; \theta)$ of text x itself (details in §3) and using this probability to predict y , reducing or obviating the need for large supervised datasets. In this section we lay out a mathematical description of the most fundamental form of prompting, which encompasses many works on prompting and can be expanded to cover others as well. Specifically, basic prompting predicts the highest-scoring \hat{y} in three steps.