Machine Learning for Procedure Learning from Instructions

Akanksha Gupta* Yuhan Shen† Ehsan Elhamifar‡

ABSTRACT

We explore and compare the different Natural Language Processing models to generate word embeddings and develop weakly supervised machine learning algorithms on these embeddings for extracting the grammar of complex tasks from instructional video data by using the narration of its audio input. By exploring the audio narration, we find interesting facts about its relation with the video as the speaker tends to narrate the step while carrying it out; they might give a precise note, or elaborate the step by talking about different scenarios, or talk about the step before carrying it out, or after the step is executed. The challenge we try to address is narrowing down the audio narration to spot the key steps used in the video to carry out a task. We explore the existing natural language processing models and localize the key steps in a video by computing cosine similarity of the sentences in audio narration with the task-related key steps. Followed by building a neural network on top of the word vectors, we show the effectiveness of neural networks to localize key steps in video based on the audio narration data and how it overruns the performance of the simple word representations of the narration text.

1 INTRODUCTION

Understanding the key steps from an instructional video to carry out a task is a complex problem. Training a network to localize the key steps requires humongous amount of data. Extracting this information from a new video given the annotations can be achieved with procedure learning. Procedure learning can be used to train autonomous agents perform complex tasks, or help humans identify key steps, or build huge knowledge bases of instructions. Extracting features from a video is a challenging task. Identification of key steps from a video can be decomposed into multiple sub-problems. Key steps can be extracted by understanding the grammar of the visual data sans the audio, and vice-versa, followed by combining the results from both audio and video. It is imperative to study both the audio and visual data and the similarities and differences between the two. A person could be talking about various possibilities to achieve a task, or laying out general principles for achieving one, or the possible outcomes from carrying out a particular key step, or different alternatives to the current approach being carried out in the video. So, it becomes important to filter the desired content that truly represents the key steps required to perform a task. Understanding the audio narration without using the visual cues makes it harder to extract the mapping between the two since the natural language can be altered just by a few words to represent different scenarios. Even defining the key steps before studying the narration might not always solve the problem since a model can study a few words or sentences at a time. Building context while studying this model can help in achieving a better outcome, otherwise distinguishing a humorous sentence in a plain narrative might not be possible.

*e-mail: gupta.aka@northeastern.edu
†e-mail: shen.yuh@northeastern.edu
‡e-mail: e.elhamifar@northeastern.edu

To explore the audio data has many possibilities: training a model on the audio input directly, or converting the audio to narration and build models on the textual data. The audio input is expected to have a lot of noise which needs careful filtering to extract the real information from a video. Running models directly on the audio signals caters to noise a lot more than expected. Audio data can be cleaned in a better way when the audio input is converted first to the textual narration after carefully examining and filtering the results and then refined again by cleaning the textual narration. Text can be studied separately while also working on the visual data. Running models on the textual narration of the audio data provides interesting key sights in understanding the instructional videos. The person in video tends to narrate the task while carrying it out, so it can provide a crisper result when combined with a model running on the visual data as compared to a model running only the visual data. However, narrations and visual data might have some misalignment, in that, the speaker sometimes misses to put the task in clear words while putting the task in action, or introduces the steps before actually carrying those out, or vice-versa. So, interpreting the natural language of the narration to localize the key steps in the video is a very challenging task but equally critical in procedure learning.

Extracting grammar from natural language has always been an interesting problem. A neural network model can be trained to learn word associations from a large corpus of text, detect synonymous words, or capture the context of a word in a document. It is an important task to identify the right type of model that fits in a given scenario. The Natural Language Processing models can be broadly divided into *context-free* and *contextual* models. Both types of models cater to different utilities. A context-free model takes one word from the document at a time and generates a vector for the word, so a single word representation is generated for each word in the vocabulary, no matter what context it is being used in. For example, Word2Vec[8,9] and GloVe[1] can be used to detect similarities mathematically, providing a good tool for building a recommendation system. Contextual models, on the other hand, build a word representation depending on the context where that word occurs, meaning that the same word in different contexts can have different representations. For example, ELMo[7,11] and Bert[4,5] can be used to generate contextual word embeddings to build a good search engine.

2 NEURAL NETWORK

A neural network is a set of algorithms represented as a network of artificial neurons that are designed to recognize patterns. Popular real world applications for neural networks are the categorization of image content, recognition of handwriting or speech and even prediction of developments in financial markets. Neural networks can adapt to changing input, hence improving the learning graph as the input data is fed into the model. Within networks, self-learning arising from experience can draw conclusions from a complex and seemingly unrelated collection of data. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

3 Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence that deals with linguistics, with the aim of enabling interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The computers can then accurately extract information and insights from documents as well as categorize and organize the documents. Most NLP techniques rely on machine learning to derive meaning from human languages. Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural-language generation.

3.1 Word2Vec

Word2vec[2] is a technique for natural language processing. The Word2Vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.

3.2 ELMo

ELMo is a deep contextualized word representation that models both (1) complex characteristics of word use such as syntax and semantics, and (2) how these uses vary across linguistic contexts. These word vectors are learned functions of the internal states of a deep bidirectional language model, which is pre-trained on a large text corpus. They can be easily added to existing models and significantly improve the state of the art across a broad range of challenging NLP problems, including question answering, textual entailment and sentiment analysis. ELMo representations are contextual and the representation for each word depends on the entire context in which it is used. The word representations combine all layers of a deep pre-trained neural network.

3.3 Bert

Bidirectional Encoder Representations from Transformers[3] is a Transformer-based machine learning technique for NLP pre-training. BERT is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. Pre-trained representations can either be context-free or contextual, and contextual representations can further be unidirectional or bidirectional. Context-free models such as word2vec or GloVe generate a single word embedding representation for each word in the vocabulary. For example, the word 'bank' would have the same context-free representation in 'bank account' and 'bank of the river.' Contextual models instead generate a representation of each word that is based on the other words in the sentence. For example, in the sentence 'I accessed the bank account,' a unidirectional contextual model would represent 'bank' based on 'I accessed the' but not 'account.' However, BERT represents 'bank' using both its previous and next context — 'I accessed the ... account' — starting from the very bottom of a deep neural network, making it deeply bidirectional.

3.4 GloVe

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates the logarithm of ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well.

4 Experiments and results

Analyzing the audio narration of the instruction videos involves using Natural Language Processing and Neural Networks. We first prepare the dataset and start training our models on this dataset. We localize the key steps by using different NLP models and find out which model fits best for this use case. Next, we move on to enhancing the results by designing a neural network on word embeddings produced by the NLP models. Lastly, we compare and show how the results improve significantly with the latter enhancement, i.e., by using neural network on the word embeddings produced by the NLP models extract deeper information and yields a better performance.

4.1 Dataset

Building a model on the video narration that spans across multiple instructional tasks requires a good deal of instructional video dataset with carefully collected and cleaned audio narration. We use Procedure Learning (ProceL) dataset[6] for research on the audio narration. The ProceL is a medium-scale dataset of 12 diverse tasks, such as perform cardiopulmonary resuscitation (CPR), make coffee and assemble clarinet. Each task consists of about 60 videos and on an average contains 8 key-steps. Each task has a grammar of key-steps, e.g. 'perform CPR' consists of 'call emergency', 'check breathing', 'check dangerous', 'check pulse', 'check response', 'open airway', 'give compression' and 'give breath'. Each video is also annotated with the key-steps based on the visual data; each video may not contain all the key steps and the order of key steps in some videos might be different, as there are multiple ways to perform the same task.

4.2 Annotations

Given the key steps for a task and also for each video for the task, we annotated each sentence from the narration with relevant key steps, if present in the sentence. Some complex sentences covered multiple key steps while some of the sentences did not cover any key steps. For example, in 'perform CPR', the sentence 'start the CPR cycle 30 compressions 2 breaths' covers two key steps 'give compression' and 'give breath', while the sentence 'you can't help anyone if you become a victim too' covers no key step. We did this for all sentences in the 60 videos of 'make coffee' and 'perform CPR' tasks. For rest of the tasks, we covered 10 videos for each task, annotating every sentence that occurs in these videos.

Sentences were annotated with relevant key steps by taking one sentence at a time, picked exclusively. That is to say that while annotating a sentence, its adjoining sentences were not considered to add a context to the current sentence. While some sentences specify one or more key steps explicitly, others provide a hint to one of the key steps. For example, 'tilt the head back lift the chin and give two breaths' is clear to talk about 'give breath' key step while 'so checking for circulation in a victim' is a little ambiguous, in that it could be mentioning either the 'check breathing' or the 'check pulse' key step or both. We distinguish the two with two different labels 'confident' for the former and 'weakly confident' for the latter, trying to mark as many labels with 'confident' as possible and narrowing down the occurrence of 'weakly confident' steps in the

Table 1: Similarity between key steps obtained from video and audio narration

Task	Levenshtein	Damerau Levenshtein	Jaro
Assemble Clarinet	0.571	0.577	0.944
Change Iphone Battery	0.545	0.554	0.919
Change Tire	0.440	0.440	0.919
Change Toilet Seat	0.504	0.512	0.918
Jump Car	0.550	0.560	0.917
Make Coffee	0.936	0.937	0.988
Make Pbj Sandwich	0.653	0.653	0.926
Make Salmon Sandwich	0.720	0.720	0.918
Perform CPR	0.930	0.931	0.984
Repot Plant	0.727	0.727	0.924
Setup Chromecast	0.799	0.799	0.924

text. This provided us a cleaner dataset to work with the confident key steps and establish a model that works with good accuracy, with a possibility of improvement by extending the training with both types of labels.

4.3 Video / Audio narration key steps comparison

We have briefly noted that there might be some expected discrepancy between the video and its audio narration. The reasons being, the speaker first hints at a few key steps and talks about those before actually carrying those out in the video, or silently does a part of the work followed by making a small note about its benefits, or speaks about two key steps one after the other while performing the two steps in the reverse order. There could be many other possibilities as to why the audio narration might not precisely narrate the tasks as being carried out in the video. Without walking through any/each of the reasons behind a discrepancy, we study the extent to which the two differ in laying down the key steps required to perform a task.

We calculate similarity between the key steps arranged by studying the video and the key steps arranged by annotating the audio text for each of the videos in a task and take average across all videos in each task. We continue to study all the 60 videos each for 'make coffee' and 'perform CPR' tasks, and 10 videos for rest of the tasks. We study similarity between the order of key steps that occur in a video vs the audio narration text, and the extra/missing key steps that occur in either one of the two. For this comparison, we calculate similarity by using edit distance[10] between the two sequences of key steps. For calculating edit distance for a single video, we get an ordered list of key steps that occur in a video and remove the consecutively repeated key steps if any. We compute the second ordered list for the key steps that occur in the textual narration data in a similar way. We use the below equation to compute similarity from these edit distance values.

$$\text{Similarity, } S = 1 - \left(\sum_{i=1}^{n} (EditDistance(x_i, y_i)/max(len(x_i, y_i))/n\right)$$

where n = number of videos in a task,

x_i = list of steps from the visual data in video i

y_i = list of steps from the textual narration for the video i

$S \in [0,1]$; 0 means totally different and 1 means identical.

We compute similarity between the key steps obtained from the video and the audio narration for each task by using three types of edit distances: Levenshtein, Damerau Levenshtein and Jaro. *Table* 1 shows how Jaro distance yields the best similarity results for each task. The results depict that the results we obtain from annotating the audio narration text are not very far from the ones that we obtain from the visual data.

4.4 Finding key steps for each sentence

Now that we have manually generated the key steps for the textual narration data and compared that these key steps are mostly in line with the ones we get from the visual data, we start to build a model that identifies the key steps covered in a sentence given the sentence as an input. We want to make sure that a key step is correctly mapped to a sentence that conveys an action towards carrying out the key step. This would provide a higher probability that the key step is being carried out in the video as well because our ultimate goal is to localize the key steps in an instructional video.

4.4.1 Word embeddings

Since a computer works best with numbers, we transform the words into relevant word embeddings. A word embedding is a mapping of a variable or a word to a vector with continuous numbers which can be used to extract the features of the text. Words with same meaning are given a similar representation in the n-dimensional space. Finding word embeddings is another field of research closely related to this task. An ideal word embedding would accurately establish similarity between the words, semantic relations, meaning and context of the words. In context-free word embedding the representation of a word is determined irrespective of the meaning of the word in a particular sentence. So a single word representation is generated for each word in the vocabulary, no matter what context it is being used in. Contextual models, on the other hand, build a word representation depending on the context where that word occurs, meaning that the same word in different contexts can have different representations.

We first build embeddings for each sentence by using different NLP models. We explore both context-free and contextual models to study this problem. After obtaining word embeddings for a sentence, we study different approaches to find key steps covered in a sentence. A sentence can represent none, one or multiple key steps, as discussed previously.

Starting with the Word2Vec context-free model, we find word embedding for each sentence in each of the videos for a task. For finding word embeddings, we first clean the data by removing all punctuation and we take one word from the sentence at a time and get the 300-dimensional vector representation for the word if present in the model's vocabulary. To compute a single embedding for the sentence, we use MaxPooling to combine the vector results from each word in the sentence. This 300-dimensional vector represents one sentence and can be used to find the key-step covered by it, if any.

Next, we use the GloVe context-free model to find embeddings for all sentences in each video of a task. We use the 'glove-twitter-25' API for building the word embeddings. This model produces 25-dimensional vector for each word, yielding a 25-dimensional vector for a sentence after we use MaxPooling on vectors for all words in the sentence. Similarly, we also use ELMo and Bert contextual word embedding models in our comparison. These models take one sentence as an input at a time and generate a vector for that sentence. So, MaxPooling layer is not needed for the two contextual models. ELMo produces a 1024-dimensional vector for a sentence, while Bert produces 768-dimensional vector for a sentence.

4.4.2 Word embeddings and Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. This metric can be used to measure how similar two sentences are irrespective of their sizes. To use this approach, we compute vector embeddings for all key steps in a video using each of the four models, Word2Vec, GloVe, ELMo and Bert. Using one model at a time, we compute cosine similarity between the n-dimensional vector of a sentence and the n-dimensional vector of each key step. This way, we calculate the cosine similarity for each sentence of every video in a task.

Table 2: F1 scores from the cosine similarity model

Task	Word2Vec	Bert	ELMo	GloVe
Assemble Clarinet	0.18	0.048	0.2	0.068
Change Iphone Battery	0.09	0.048	0.12	0.058
Change Tire	0.05	0.043	0.1	0.035
Change Toilet Seat	0.07	0.029	0.2	0.069
Jump Car	0.07	0.044	0.17	0.114
Make Coffee	0.18	0.051	0.22	0.057
Make Pbj Sandwich	0.06	0.063	0.2	0.077
Make Salmon Sandwich	0.08	0.051	0.15	0.076
Perform CPR	0.5	0.108	0.2	0.113
Repot Plant	0.02	0.050	0.1	0.094
Setup Chromecast	0.09	0.034	0.1	0.044

Given a sentence s_i, a key step k_j and an embedding function $f(\cdot)$, we compute the cosine similarity between the sentence and key step as follows:

$$p_{i,j} = \frac{\langle f(s_i).f(k_j)\rangle}{\|f(s_i)\|\|f(k_j)\|}. \quad (1)$$

After that, we set a threshold t and predict that key step k_j is present in the sentence s_i if $p_{i,j} > t$. We adjust this threshold value across the tasks to obtain good results. For comparing the results, we compute F1 score, precision and recall parameters between the key steps identified by our cosine similarity model and the key steps manually annotated in the dataset for each video. These values allow us to adjust the threshold to keep the precision and recall values comparable to each other. We compare results with both the 'confident' marked labels and 'confident' and 'weakly confident' labels taken together.

Table 2 depicts that Word2Vec model performs best for the 'Perform CPR' task and ELMo model performs best for rest of the tasks. We also note that the F1 scores for Word2Vec on 'Perform CPR' and ELMo on 'Make Coffee' are the best scores among other videos. For these two tasks, we perform the experiments on the input data of 60 videos, while for the rest of the tasks, we perform experiments on the input data of 10 videos, which is why the models perform better for the two tasks when compared with the other tasks. 'Make coffee' in particular has been an interesting task to work with, because some of its key steps like 'see coffee' and 'clean pot' have been mentioned in the textual narration data in very different ways. For example, some of the sentences for 'see coffee' read as, 'so I'm gonna open the lid and check if we are brewing espresso' or 'and we'll check back in a couple of minutes', which make it hard for the model to map it accurately to this key step with a good similarity value.

While working on this problem, we observe that many long compound sentences that cover one key step miss to get labelled with that key step. The reason being that the key step is nothing but a short phrase of two to four words, so the overall vector for the sentence no longer holds a good similarity with this key step, if it is a long compound sentence. To overcome this problem, we add another enhancement to our model by splitting each sentence into sub-parts of 5 words each, with four overlapping words between the two consecutive sub-parts. As we expected, many long compound sentences get correctly mapped to the corresponding key steps. However, it gave rise to many 'False positive' sentence to key step pairs. This was because a sub-part of the sentence with only 5 words sometimes misrepresented the real context of the sentence and showed a strong correlation with the corresponding 2-word key step. This is why we do not find better results with this enhancement.

Another enhancement we add is the use of verb phrases. We define verb phrases for each sentence in every video for a task; a sentence is mapped to none/one/many verb phrases. We run the similar experiment and localize the key steps in the video by computing cosine similarities between the verb phrases of a sentence and the key steps defined for the video. We again define thresholds for each task for better performance. However, with this experiment, we get pretty low F1 scores because a verb phrase might not always convey the same meaning as the sentence, which can decrease or increase the cosine similarity with the relevant key steps. For example, the sentence 'next check the victim for breathing or only gasping' gets the verb phrase 'check the victim', which does not have a good similarity with the 'check breathing' key step, while the sentence itself has a good cosine similarity with this key step, thereby decreasing the probability of the sentence being correctly matched with the key step.

Table 3: F1 scores from Neural Network model with Word2Vec embeddings

Task	F1 score	Precision	Recall
Assemble Clarinet	0.303	0.405	0.242
Change Iphone Battery	0.265	0.361	0.210
Change Tire	0.333	0.417	0.278
Change Toilet Seat	0.314	0.455	0.240
Jump Car	0.276	0.500	0.190
Make Coffee	0.504	0.528	0.483
Make Pbj Sandwich	0.216	0.333	0.160
Make Salmon Sandwich	0.150	0.250	0.107
Perform CPR	0.834	0.880	0.792
Repot Plant	0.074	0.500	0.040
Setup Chromecast	0.437	0.514	0.380

4.4.3 Word embeddings and Neural Network

For this milestone, we use Neural Network to extract denser information from the sentence vectors to find the key steps covered in the sentence. While the NLP models study each word with/without the context and provide embeddings that help to identify similarities between words or context of a word in a sentence, Neural network can help uncover some unseen features from the dataset by training the model along with the expected output for the training dataset. This problem serves as a multi-label classification problem, because each video has multiple key steps or classes and a sentence from the video can be classified with none/one/many key-steps, making it a multi-label classification problem.

We use Dense Neural layers to produce better results from the sentence embeddings. We define the input as a vector of n-dimensions (n depending on the NLP model used) for each sentence and the output layer as a vector of probabilities of length equal to the number of key steps defined for the video. We use a dense layer of 64 units with a Rectified Linear activation to throw the negative results. We define the output layer as a dense layers with number of units same as the number of key steps in the video with a Sigmoid activation to obtain a 0-1 range probability of the occurrence of a key step in a sentence. We compile the model with binary cross-entropy loss function and Adam optimizer. Using a 70% training to test ratio, we train the model and evaluate its results. Even as the amount of training data is low, we find good results from our Neural network model.

Table 3 and *Table* 4 show the best results we obtain from our Neural Network model for each video. We run this model on the sentence vectors obtained from Word2Vec and ELMo models. The tasks 'Make Coffee' and 'Perform CPR' again outperform the rest of the tasks for the same reason i.e. because we have 60 videos data for each of the two tasks as compared to 10 videos data for the other tasks. We also observe from *Table* 2, *Table* 3 and *Table* 4 that for each task, the F1 score obtained with Neural Network layers is much better as compared to the F1 score obtained with the previous approach of localizing the key steps based on cosine similarity and a threshold value. We get good results by using ELMo embeddings for

Table 4: F1 scores from Neural Network model with ELMo embeddings

Task	F1 score	Precision	Recall
Assemble Clarinet	0.318	0.538	0.226
Change Iphone Battery	0.326	0.5	0.242
Change Tire	0.433	0.542	0.361
Change Toilet Seat	0.367	0.628	0.260
Jump Car	0.357	0.625	0.250
Make Coffee	0.538	0.606	0.483
Make Pbj Sandwich	0.158	0.231	0.120
Make Salmon Sandwich	0.222	0.294	0.179
Perform CPR	0.772	0.826	0.725
Repot Plant	0.118	0.222	0.080
Setup Chromecast	0.4	0.533	0.320

most of the tasks, except for 'Perform CPR', 'Make Pbj Sandwich' and 'Setup Chromecast' tasks, where Word2Vec embeddings yield better results.

5 Future Work

The dataset contains more number of negative samples as compared to the number of positive samples. This demeans the performance of the network since the network gets lesser training on the positive samples and hence inclines more towards marking the key steps with much lower probabilities than expected. This problem can be overcome by reducing the number of negative samples in the input data. In order to achieve this, the input can be changed from an n-dimensional vector per sentence to a pair of sentence and key step with a probability that the pair belongs to a positive sample or a negative sample. Some of the negative samples can be dropped from the training dataset to achieve a ratio of 1:5 or 1:10 for the number of positive to negative samples.

A custom loss function can be defined to cater to this improved dataset. Hinge loss could be used with the input training set marked with -1 for a negative sample and +1 for a positive sample. In addition to this, training can be performed on all the tasks taken together to make the model more robust. With this model, given a new video with new key steps for a new task, the same model would be able to localize the key steps in the new task video without being trained separately for this new task.

Instead of using existing embedding function such as Word2Vec, ELMo, we can train a neural network to obtain the embedding function $f(\cdot)$ for our prediction.

6 Conclusion

We develop a neural network on the sentence vector embeddings and show how the neural network model performs a better task in localizing the key steps in an instructional video's textual narration data as compared to a model that localizes the key steps based on cosine similarities of the sentence/key step vector embeddings.

References

[1] https://nlp.stanford.edu/projects/glove/.

[2] https://en.wikipedia.org/wiki/Word2vec.

[3] https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

[4] J. Devlin and M.-W. Chang. Open sourcing bert: State-of-the-art pre-training for natural language processing. *Google AI Blog. Weblog.[Online] Available from: https://ai. googleblog. com/2018/11/open-sourcing-bertstate-of-art-pre. html [Accessed 4 December 2019]*, 2018.

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

[6] E. Elhamifar and Z. Naing. Unsupervised procedure learning via joint dynamic summarization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6341–6350, 2019.

[7] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. H. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer. A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2017.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119, 2013.

[10] G. Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.