Capsule Networks for NLP

William Merrill

November 2018

1 Introduction

Capsule networks are a sophisticated neural network architecture originally proposed in the domain of computer vision. Motivated by some of the main problems with convolutional neural networks for image classification, capsule layers allow low-level features about an image to be effectively combined into higher-level information. The inherent compositionality begs the question of whether capsule networks might also be useful for natural language processing tasks - specifically, text classification. In my presentation, I first introduce the capsule network architecture, and then discuss three recent papers which apply it to different variants of text classification.

Convolutional neural networks are a standard architecture for many computer vision tasks, but they also suffer from many shortcomings. For example, they struggle to learn transformational invariance of images. Additionally, the presence of max pooling layers can make it hard for the model to ensure that there is local agreement between features.

The capsule network aims to remedy these problems by replacing the scalar-valued filters in a convolutional neural network with vector-valued capsules. Each capsule encodes a particular feature that the network is looking for. By design, the magnitude of a capsule vector represents the probability of the existence of that feature at a particular position (this is analogous to the value of a filter in a convolutional network). The direction of a capsule vector can then encode parameters about the object that is being detected. For example, a capsule that is detecting lines might point one way if the line is red and another way if the line is blue.

To connect layers, capsule networks compute a learned vote vector between each pair of connected capsules that encodes what information is relevant to be passed up to the next layer. Then, a parameterless algorithm called dynamic routing is used to compute the value of the the capsules in the next layer in terms of the previous layer's vote vectors. This sophisticated method of connecting layers replaces max pooling, and it is designed so that the connections between capsules can enforce the agreement of features in the previous layer.

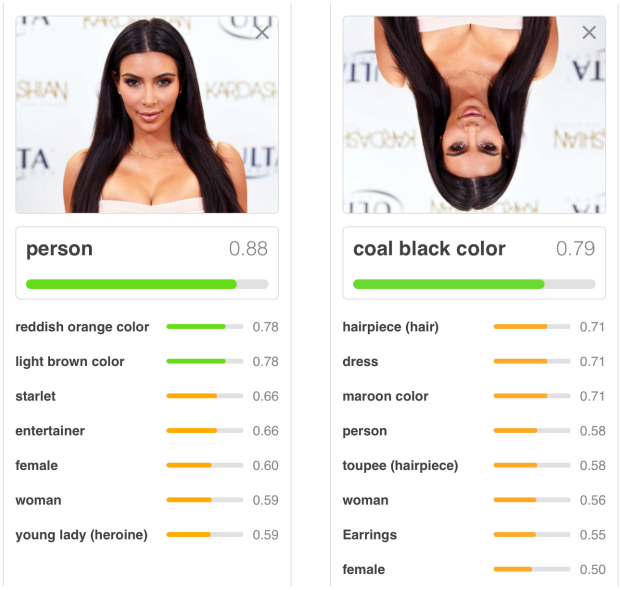

Figure 1: Turning a face upside down fools a state-of-the-art classifier. From https://medium.freecodecamp.org/understanding-capsule-networks-ais-alluring-new-architecture-bdb228173ddc.

2 Keywords

convolutional networks, capsules, vote vectors, dynamic routing, margin loss, text classification, capsule networks

3 Background material

3.1 Problems with Convolutional Neural Networks

A convolutional neural network is a cascade of convolutional layers and intermittent max-pooling layers. The convolutional filters have the effect of picking out important patterns in local neighborhoods across the image, and the pooling layers have the effect of reducing the size of this data to include only that which is most relevant. After the series of convolutions, we flatten all of our filters into a feature vector and use one or more feed-forward layers to perform classification.

One major problem with this architecture is its inability to learn good kinds of transformational invariance. As we can see in Figure 1, it is very difficult to learn to correctly classify an upside down face because we essentially need to learn a different filter for each rotation of the face. This means we will need a massive amount of training data if we want our network to deal with rotations and flips properly.

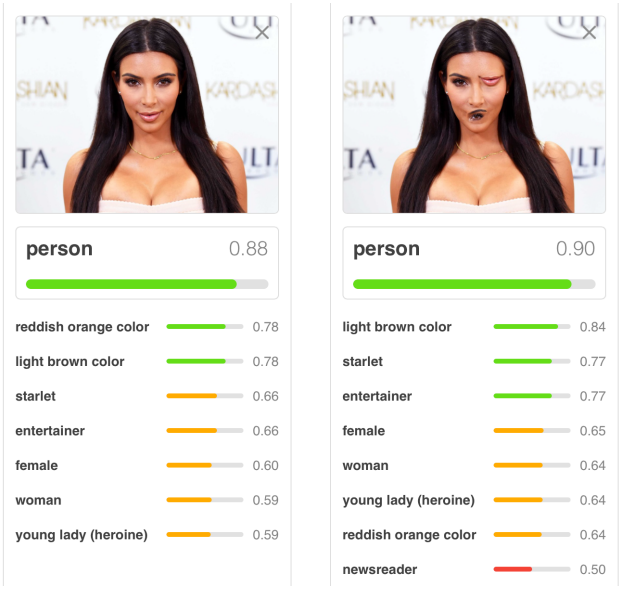

Figure 2: Max pooling loses information about the relative position of features. From https://medium.freecodecamp.org/understanding-capsule-networks-ais-alluring-new-architecture-bdb228173ddc.

Let $v \in \mathbb{R}^d$ be a capsule.
$\|v\| \in (0, 1)$ is the probability that the feature detected by the capsule exists.
The direction of v encodes instantiation parameters.

Figure 3: Semantics of the capsule representation Sabour et al. [2017].

A related problem is the fact that important information about the relevant position of lower-level features is lost by max pooling layers. For example, in Figure 2 we have a mouth near a nose in an input image. A pooling layer then collapses these features to occur at the same pixel. At this point, it is impossible for the next layer to know whether the noise occured above the mouth or vice versa, which is extremely relevant as to whether the features induce a face. Therefore, we would like our model to be sensitive to this kind of local "agreement" information.

3.2 Capsule Network Architecture

Figure 3 provides further details about the capsule representation. Figure 4 shows the dynamic routing algorithm which computes the value of capsules in the next layer from the preceding layer's vote vectors. Figure 5 shows the form of the squash function that is used by this routing process. Finally, Figure 6 defines the max-margin loss that can be used to train capsule networks.

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$

Figure 4: The dynamic routing algorithm Sabour et al. [2017].

$$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|} \tag{1}$$

Figure 5: The capsule squash function Sabour et al. [2017].

4 Presentation of papers

4.1 Investigating Capsule Networks with Dynamic Routing for Text Classification [Zhao et al., 2018]

This paper was the first work to apply capsule networks to a text classification task. They evaluate the performance of a capsule network classification model on a variety of single-class and multi-class classification tasks. The multi-class data sets have single-class training data and multi-class evaluation data, so to perform well, the model must learn to transfer knowledge effectively from the single-class to multi-class case.

The architecture used by this paper provides a starting point for the architectures used by the later papers. The first layer of the network is a standard convolutional layer, which acts as a feature extractor. The next layer is a primary capsule layer, which is a kind of layer that maps scalar-valued features into a capsule representation. After this, these capsules are fed through a convolutional capsule layer. Finally, the output of the convolution is flattened and fed through a feed-forward capsule layer. Each capsule in the output of the this layer is interpreted as corresponding to a particular class. An additional orphan capsule that corresponds to no class is also present in the final layer. The model

$$L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda(1-T_k)\max(0, \|v_k\| - m^-)^2 \tag{2}$$

Figure 6: Max-margin loss, where T_k is an indicator function for class k. Note that m^+ and m^- are constants Sabour et al. [2017].

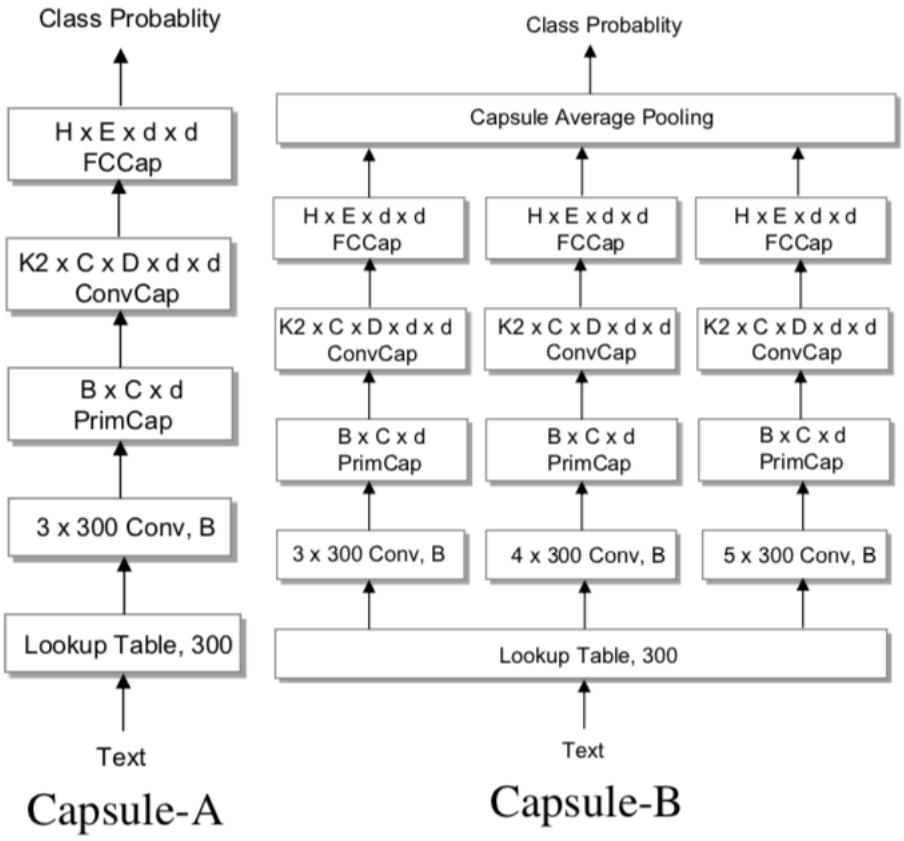

Figure 7: The architectures used by Zhao et al. [2018].

is trained using max-margin loss. The authors also experiment with a variant where three independent capsule networks are combined via averaging at the end. The two architectures are illustrated in Figure 7.

The results compare the performance of this capsule architecture to many different strong baseline methods. In single-class classification, the standard capsule network architecture performs comparably to the best baseline methods. The authors also try a three-headed variant of their capsule network where three network computations are performed in parallel and then averaged, and this performs marginally better than the baselines. In the multi-class case, both capsule methods outperform the baselines, with the greatest increases again reported for the three-headed network. This suggests that the capsule representation is very effective at transferring knowledge from the single-class to multi-class problem.

4.2 Identifying Aggression and Toxicity in Comments using Capsule Network [Srivastava et al., 2018]

This paper focuses on a specific application of text classification: namely, classifying toxic comments. They look at two data sets for this task: one of comments in English, and the other which mixes English and Hindi. They hypothesize that the capsule architecture will be particular helpful in the code-mixed case because the local representations built up by a capsule network should handle mixing better than sequence models.

The authors approach this task using an architecture similar to the one used by Zhao et al. [2018]. They make two modifications to the architecture. First, they use an LSTM for the initial feature extraction layer instead of a convolutional network. Second, they replace the final feed-forward capsule layer with a standard feed-forward layer. The reason for this is so that they can use a focal loss function, with which they hope to overcome the class imbalance problems inherent to toxic comment classification.

Again, the authors compare their capsule architecture to a variety of competitive baseline methods. On both data sets, the capsule architecture outperforms the baseline methods. On the English data set, they achieve 98.46% accuracy, which is roughly 0.2% greater than the next best method. On different versions of the code-mixed data set, they report accuracies of 63.43% and 59.41%, which in both cases constitutes an increase of roughly 1% over the next-best method. These results imply that the capsule architecture is especially effective at classifying toxic comments when they are code-mixed.

4.3 Zero-shot User Intent Detection via Capsule Neural Networks [Xia et al., 2018]

This paper applies a similar capsule method to the classification task of zero-shot intent detection. Standard intent detection is the task where a dialog system must classify the an actionable intent that a user's query represents from a known set of intents. For example, "Play a song!" might be classified as PLAY-MUSIC. The zero-shot problem is more complicated: in this case, we have access to a set of known intents during training, but, at inference time, our goal is to classify from a set of previously unencountered intents. For example, imagine that we already trained a model on data containing the PLAY-MUSIC intent, but our model has never seen PLAY-MOVIE. The goal is that we can still map "Play a movie!" to PLAY-MOVIE. Doing this requires the model to generalize what it has learned about the known intents to the unknown ones.

The authors devise an architecture that consists of three modules: two which are present during training and whose output capsules are a prediction over known classes, and a third one which attempts to generalize from these intermediate output capsules to unknown intents. The first two modules are rather similar to the architecture used by Zhao et al. [2018], except that the feature extraction layer is an LSTM with self-attention. The major architectural difference is the third untrained module for zero-shot inference. First, the authors

compute a similarity matrix between known intents and unknown intents by using word embeddings for the words in the intent names. Then, they compute new vote vectors for the zero-shot layer by taking a weighted average of the vote vectors for the layer classifying known intents. They then use standard dynamic routing to convert these vote vectors into a layer of classification capsules for the unknown intents.

The experimental results show that the known intent detection module by itself outperforms baselines for known intent detection across two different data sets. However, the key result is that the zero-shot intent detection network significantly outperforms the previous baseline systems against which it is compared. Thus, the authors are able to effectively utilize the capsule network architecture to do zero-shot inference.

5 Comparison of Papers

Each paper applies a capsule network architecture to some variant of a text classification task. In the case of Zhao et al. [2018], the capsule model is applied to a variety of single-class and multi-class text classification tasks. Srivastava et al. [2018], on the other hand, focus specifically on classifying toxic comments. Finally, Xia et al. [2018] focuses on the the classification task of intent detection, both in the standard and zero-shot case.

Viewed together, these papers develop a standard capsule network architecture that can be used for text classification tasks:

1. A feature extractor layer.
2. A primary capsule layer.
3. A convolutional capsule layer.
4. A classification layer.

In each paper, the capsule network performs at or above the previous state of the art. Zhao et al. [2018] show how their capsule model is able to effectively transfer knowledge from single-class classification to multi-class classification. Similarly, Xia et al. [2018] show how their capsule model does a good job at transferring knowledge from standard intent detection to zero-shot intent detection. Srivastava et al. [2018] find that capsule networks are especially strong at dealing with code-mixed text.

6 Summary

In this chapter, I first introduced the underlying mechanics of the capsule network architecture. I then reviewed three recent papers which apply this network to variants of text classification. Each paper shows how a capsule network performs comparably to or better than the previous state of the art on a text

classification task. Additionally, the papers leverage the representational power of capsules for effective transfer learning. The potential for transfer learning might be one of the most attractive things about the capsule network architecture for NLP tasks. Another direction for future work is the adaptation of capsule networks to NLP tasks besides text classification.

References

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

Saurabh Srivastava, Prerna Khurana, and Vartika Tewari. Identifying aggression and toxicity in comments using capsule network. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 98–105, 2018.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. Zero-shot user intent detection via capsule neural networks. *arXiv preprint arXiv:1809.00385*, 2018.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*, 2018.