Named Entity Recognition with Capsule Network

Matthew Potts UC Berkeley
Dave Huber UC Berkeley
Legg Yeung UC Berkeley
{mepotts, dhuber, yeunghoman}@berkeley.edu

Abstract

Dynamic Routing Between Capsules (Sabour et al., 2017) showed the Capsule Network (CapsNet) architecture can achieve state of the art performance on MNIST classification. A simple 3-layer CapsNet gives noticeably lower test error than a standard convolutional neural network (CNN) with similar depth. We posit that the mathematical and architectural properties that make CapsNet superior for digit recognition can extend to named entity recognition (NER). Keeping the network architectures close to that of Sabour et al., we compare CapsNet against CNN on the CoNLL-2003 Shared Task. Our 3-layer CapsNet achieves an F_1 rate of 0.874%, compared to CNN F_1 rate of 0.861%. An in-depth error analysis shows common weaknesses and strengths of the models, and that CapsNet is more sensitive to token boundaries in multi-word entities than CNN.

1 Introduction

For both computer vision and natural language processing tasks, CNNs have been widely used in detection and recognition tasks because they are fast to train and able to extract implicit features automatically. For image classification tasks, CNNs can abstract pixel level intensity values to multiple feature maps that describe properties such as edges and shades. Similarly, for natural language processing tasks, CNNs have been used to abstract word-level embeddings to multiple feature maps that capture implicit N-gram level signals. These feature maps are then fed into several feed-forward layers with ReLU non-linearity, before a softmax classifier is applied to generate normalized probabilities corresponding to a set of target classes.

In *Dynamic Routing Between Capsules*, Sabour et al. suggested CapsNet as a better alternative to CNNs for image entity recognition tasks. Both CapsNet and CNNs can take 2-dimensional values of multiple color channels as input and make predictions for the most likely target class. For MNIST digit classification, keeping the network depth similar, CapsNet achieved a test error of 0.25% compared to 0.39% for a standard CNN.

This result begs the question of whether similar performance gain can be achieved on NER tasks. Three major advantages of CapsNet over CNNs motivated the experiments in *Dynamic Routing Between Capsules*. First, a capsule's vector output in CapsNet can retain useful information about pose (position, orientation and size) better than the scalar max-pooling output in CNNs. Second, dynamic routing in CapsNet can better direct lower level capsule outputs to higher level capsules (e.g. entities) by agreement. Third, because each lower level capsule can route their outputs to multiple higher level capsules, CapsNet can recognize highly overlapping entities. Since NER tasks can benefit from retaining more word-level information such as position and tags, composing multiple-word entities from adjacent words, and understanding competing entity tags for any particular word, we posit that CapsNet intuitions can be transferred to replace CNNs in NER tasks as well.

Our main contribution lies in extending and validating this CapsNet over CNN intuition for an NER task. We compare these 2 networks using the CoNLL-2003 dataset, and provide a walkthrough of our CapsNet models in terms of architecture, test evaluation and an extensive error analysis, which shows that CNN and CapsNet are just as limited as window models, and both are capable of inferring entity classes for out-of-vocabulary words. While CNN may over-extend entity boundaries because of max-pooling, CapsNet may end entities prematurely because of the nuance information and thus over-sensitivity stored in capsule states. We suggest refining the number of iterations for dynamic routing in a future exploration.

2 Background

The formulation of our experiment is inspired by earlier research on named entity recognition in natural language processing and digit recognition in computer vision.

2.1 Dataset

We compare our models by their NER performance on the CoNLL-2003 English corpus, which was extracted from Reuters news stories between August 1996 and August 1997. Table 1 and Table 2 provide brief summaries of the text units and named entity classes. The CoNLL-2003 Shared Task on this benchmark dataset was attempted by 16 teams of researchers using a wide range of modeling paradigms including Maximum Entropy Models, Hidden Markov Models, Support Vector Machines, Conditional Random Fields, Neural Nets and combined models. Our experiment instead focuses on CapsNet and CNN. In terms of input features, each line in each data file contains four fields: the word, its part-of-speech tag, its chunk tag, and its named entity which is the target. The research teams in the CoNLL-2003 Shared Task tended to use extra information such as gazetteers, unannotated data and externally developed name entity recognizers. In our experiment, because a large number of parameters have to be trained for CapsNet and CNN, we limit our input features to word-level embeddings, part-of-speech tags and word capitalizations. In the The CoNLL-2003 Shared Task[1], the lowest performing LSTM model (Hammerton, 2003) achieved a precision of 69.09%, recall of 53.26%, and F_1 rate of 60.15%. The highest performing combined classifier (Florian et al., 2003) achieved a precision of 88.99%, recall of 88.54%, and F_1 rate of 88.76%. We adopt the same performance metrics to compare CapsNet and CNN, and observe that our best performing CapsNet would have ranked third while a baseline CNN of the same depth would have ranked fourth in the CoNLL-2003 Shared Task.

[1]The baseline predictions were produced by a system which only identifies entities for tokens which have a unique named entity label in the training data. In other words, this system memorizes the tokens with real named entities it has seen in the training set and applies its memory on the development and testing data. If a token was part of more than one named entity, then the system would select the named entity that spans the longest number of tokens. This simple baseline has a precision of 71.91%, recall of 50.90% and F_1 rate of 59.61% – less than one percent short of the lowest performing of the 16 models in the CoNLL-2003 Shared Task.

English data	**Articles**	**Sentences**	**Tokens**
Training set	946	14,987	203,621
Development set	216	3,466	51,362
Test set	231	3,684	46,435

Table 1: Number of articles, sentences and tokens per data file from the CoNLL-2003 English dataset. Directly referenced from (Table 1) in *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition* (Tjong Kim Sang and De Meulder, 2003)

English data	**LOC**	**MISC**	**ORG**	**PER**
Training set	7140	3438	6321	6600
Development set	1837	922	1341	1842
Test set	1668	702	1661	1617

Table 2: Number of named entities per data file from the CoNLL-2003 English dataset. Directly referenced from (Table 2) in *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition* (Tjong Kim Sang and De Meulder, 2003). LOC refers to location; MISC refers to miscellaneous; ORG refers to organization; PER refers to person.

2.2 State of the Art NER Model

State of the art performance on the CoNLL-2003 English dataset was achieved in *Named Entity Recognition with Bidirectional LSTM-CNNs* (Chiu and Nichols, 2015), with an F_1 rate of 91.62%. The researchers tested high-dimensional word-level embeddings randomly initialized or pre-trained from various sources[2], CNN-extracted character features, randomly initialized 25-dimensional character embeddings, capitalization features, external lexicon DBpedia and additional character features as inputs into a two-level, stacked Bidirectional LSTM. Because the convolutional, dynamic routing and decoder feed-forward layers in CapsNet require a larger number of trainable parameters relative to recurrent cells in LSTMs, our input features are not as extensive. Within our scope of investigation, we focus primarily on comparing CapsNet and CNN on the CoNLL-2003 dataset and leave the opportunity to explore hybrid architecture and extensive input features as topics for future research.

2.3 Encoding Primary Capsules

The initial success of CapsNet depended largely on the model's ability to encapsulate inverse graphics information within its internal states.

[2]Referenced from (Table 7) in *Named Entity Recognition with Bidirectional LSTM-CNNs*(Chiu and Nichols, 2015)

In *Transforming Auto-encoders* (Hinton et al., 2011), the researchers proposed computing inverse graphics from MNIST pixels using transforming autoencoders and using the generated vectors as instantiation parameters of primary layer capsules in a full capsule network. This autoencoder is composed of hidden layers and probability gates, and the intermediate states it outputs can be routed to higher level capsules, or decoded to reconstruct the original digits. Our research is partly built on this idea, but adapted to accommodate sliding windows of word token features instead of 2-dimensional image pixels. Also, we use only convolutions and a non-linear squash function to instantiate the primary capsules.

2.4 Existing CapsNet Architecture

We keep the architecture of ours CapsNet and CNN models as close as possible to those presented by Sabour et al. (2017), with the input and target formats adapted to word level representations and named entity classes found in the CoNLL-2003 dataset. The CapsNet model has one convolutional layer, two dynamically routed capsule layers and one fully connected layer, and the CNN model has three convolutional layers and two fully connected layers. Following Sabour et al., we also implement a decoder to reconstruct the input targets. This decoder acts as a regularizer to avoid over-fitting and encourages our model to retain semantic features in deeper layers of the model.

3 Methods

We established a CapsNet model and a CNN of similar depth as a baseline model for an apples-to-apples comparison.

3.1 Core Features

Our CapsNet and CNN models both use Stanford's GloVe embeddings trained on 6 billion words from Wikipedia and web text (Pennington et al., 2014). We explored both retrainable and non-retrainable word embeddings. Each word is represented by a 100-dimensional vector concatenated from 50-dimensional GloVe embedding, 45-dimensional part-of-speech features and 5-dimensional capitalization features (`allCaps, upperInitial, lowercase, mixedCaps, noinfo`). Both models ingest words in windows of length 11, with the word of interest placed at the center of the window.

3.2 Core Architecture – CapsNet

A diagram of our CapsNet architecture is provided in Figure 1. In the main model, a size-11 sliding window with the target word in the center, ingests concatenated embeddings, part-of-speech and chunk tags. `conv1` has 256 channels, each of size[3] 5x100, stride 1 and ReLU activation. This layer converts word level features into more implicit local 5-gram features. At this stage, the feature maps have shape $(7, 1)$. They are then passed into `conv2` of 32x8 channels, each of size 3x1, again stride 1 and ReLU activation. At this stage, the feature maps have shape $(5, 1)$. They are then reshaped into 160 capsules each of 8-dimensions. A nonlinear function is applied on all capsules to squash their individual vector lengths back to the $[0, 1]$ range. This process generates the primary capsule outputs.

Then, we compute these $i = 160$ primary capsule outputs to j second layer capsules outputs. This mathematical procedure closely follows that of Sabour et al. (2017). The second layer capsules each correspond to a named entity class, so there are $j = 9$ capsules where each capsule is 16-dimensional. After an inter-layer capsule prediction and a 3-iteration dynamic routing process[4], the probability that the current word of interest belongs to a particular named entity class j is now represented by the scalar length, an L2 norm, of the corresponding second layer capsule vector. We back-propagate the same margin loss[5] proposed by Sabour et al. (2017) during training. Because the CapsNet architecture allows multiple digits in MNIST recognition, we extrapolate that it also allows overlapping entities on a particular word in

[3]Embedding size is 50, 1-hot vector for PoS tags is 45, and 1-hot vector for Capitalization info is 5. Together these vectors are concatenated to a total length of 100.

[4]Adapting equations (1,2,3) from Sabour et al. (2017), this process begins with multiplying primary capsule output $\mathbf{u}_i$ by weight matrix $\mathbf{W}_{ij}$ to give prediction vectors $\hat{\mathbf{u}}_{j|i}$. We then compute a weighted sum of $\hat{\mathbf{u}}_{j|i}$ corresponding to each named entity class j by coupling with coefficients c_{ij} and summing over all i. This weighted sum is annotated as $\mathbf{s}_j$. Finally, $\mathbf{s}_j$ is then squashed to give the second layer capsule output $\mathbf{v}_j$. A 3-iteration routing process is used to update c_{ij}. In the first iteration, we initialize $b_{ij} = 0$ to represent the log probabilities that capsule i should be coupled to capsule j, then compute a softmax of b_{ij} to give the first iteration of c_{ij}. In the following iterations, we update b_{ij} by a delta of $\hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ to compute $\mathbf{s}_j$ and $\mathbf{v}_j$ again.

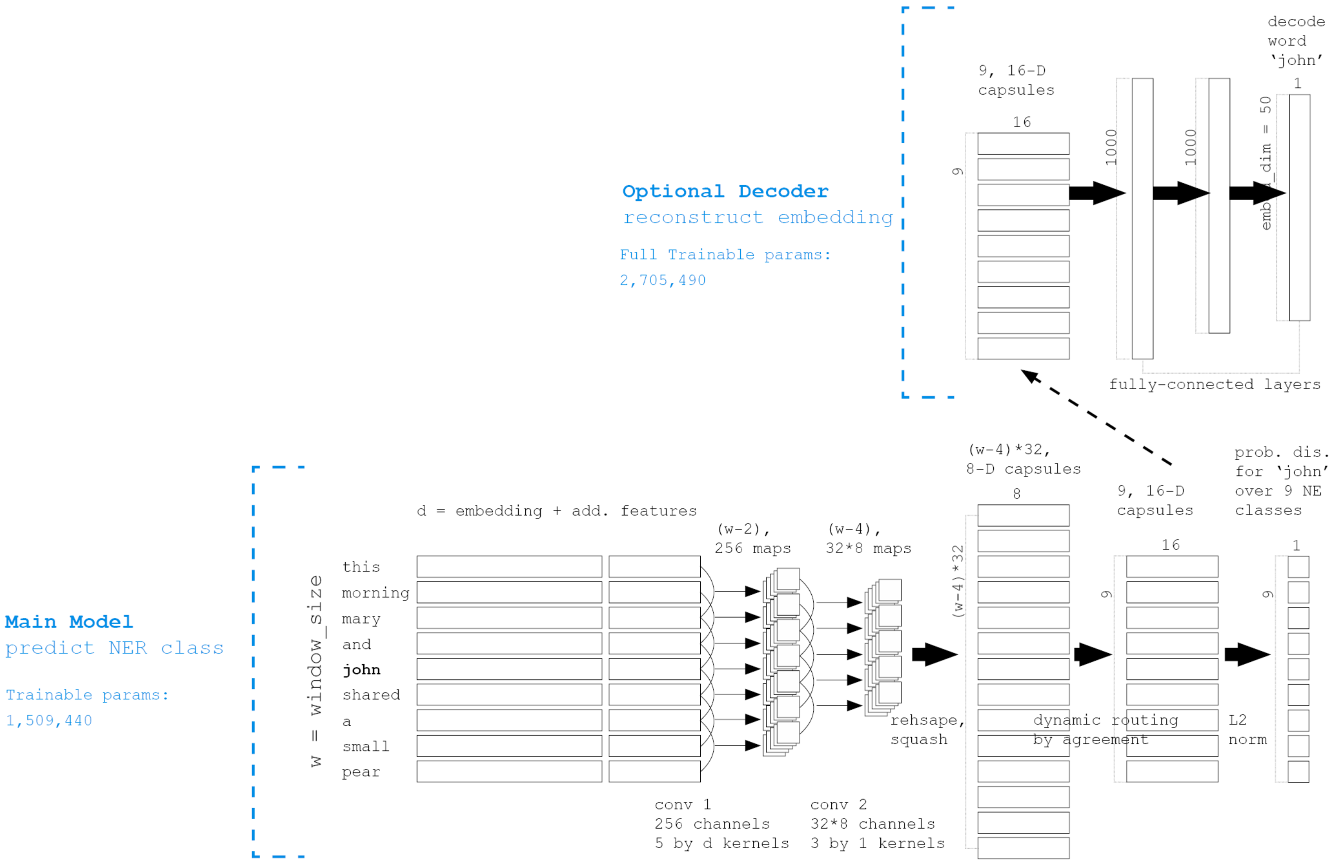
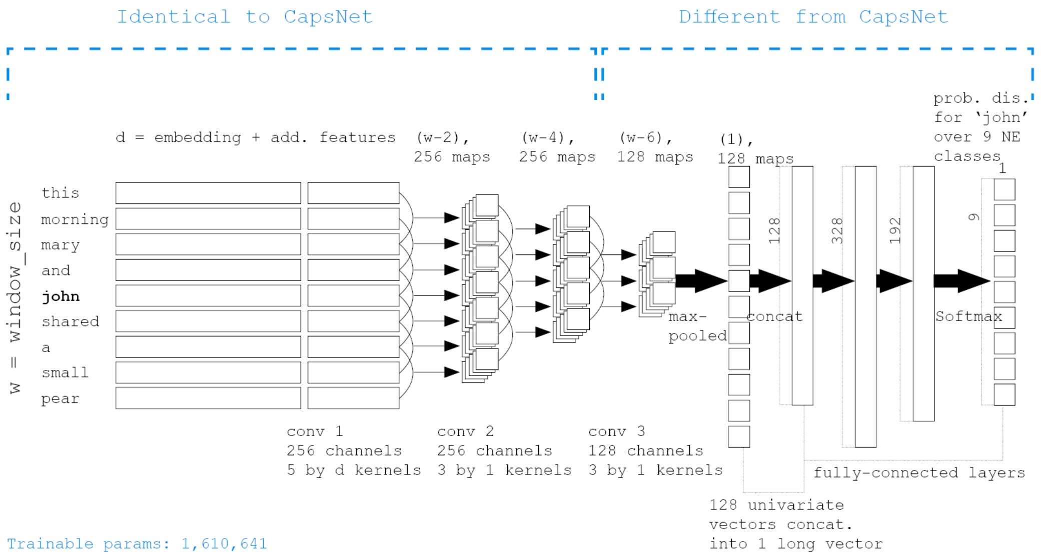


Figure 1: CapsNet Architecture for Named Entity Recognition



Identical to CapsNet
Different from CapsNet
d = embedding + add. features
w = window_size
this
morning
mary
and
john
shared
a
small
pear
(w-2), 256 maps
(w-4), 256 maps
(w-6), 128 maps
(1), 128 maps
max-pooled
concat
128
328
192
9
1
Softmax
prob. dis. for 'john' over 9 NE classes
conv 1 256 channels 5 by d kernels
conv 2 256 channels 3 by 1 kernels
conv 3 128 channels 3 by 1 kernels
fully-connected layers
128 univariate vectors concat. into 1 long vector
Trainable params: 1,610,641

Figure 2: CNN Baseline Architecture for Named Entity Recognition

named entity recognition.

As a regularization method, we implement a decoder structure that reconstructs the target word, the center word of the input window. Our decoder flattens the final capsule layer outputs into a long vector of length 144, and then masks it by the gold named entity label at training or predicted named entity label at prediction. The resultant vector propagates through two feed-forward layers to reconstruct the 50-dimensional embedding of the target word. Cosine proximity loss is backpropagated during training. This decoder can be turned off optionally. The main model has a total of $1,509,440$ trainable parameters, and the full model with decoder has $2,705,490$ trainable pa-

[5]Adapting equation (4) from Sabour et al. (2017), margin loss L_k for digit class k is adapted to named entity class k. Total loss per example sums L_k over all 9 NER classes.

rameters.

Our models were implemented using Keras. The implementation was first forked from, and then modified on top of another implementation[6] that replicated the MNIST experiments of Sabour et al. (2017). It was written and made available by Xifeng (2017)

3.3 Baseline Architecture – CNN

A diagram of our baseline CNN architecture is provided in Figure 2. The architecture is a standard CNN with three convolutional layers. All layers up to `conv2` are identical to the CapsNet. We switch out the construction of capsules and dynamic routing layers for `conv3` (128 channels, size 3, stride 1 and ReLU activation), a max-pooling layer and two feed-forward layers of size 328 and 192. The output state is then connected to a 9-class softmax layer, and cross-entropy loss is back-propagated during training. In total, this model has $1,610,641$ parameters.

4 Results

We performed experiments on model architecture using the training and development sets before evaluating three selected models on the test set.

4.1 Architecture Experimentation

We experimented with various architectural features on the development set, including the decoder, 3 window sizes, re-trainable[7] GloVe or randomly initialized embeddings, and features additional to word embeddings. The major hyper-parameters we considered include convolution kernel sizes, number of channels, hidden layer sizes and dropout rates. Of the forty-two best performing models on the development set, four have decoders; eight have window size 11 rather than 9 or 7; all use re-trainable embeddings; four use part-of-speech and capitalization features. CapsNet models make up all of the top twenty-three models rather than CNNs. It seems that larger window size and re-trainable GloVe embeddings improve performance in general, while the additional parameters needed for the decoder and additional features cancels out their advantages. CNNs seem generally inferior to CapsNets in terms of F_1 rate but converges with fewer epochs. Table 3 highlights some models from this process.

#	Mod	Decode	Win	Embed	+Fea	$F_{\beta=1}$
1	CAP	off	11	glove	yes	92.24
2	CAP	off	11	glove	no	92.15
3	CAP	on	7	glove	no	92.11
24	CAP	off	9	glove	yes	90.93
26	CAP	off	11	rand.	yes	90.79
27	CNN	off	11	glove	no	90.59
33	CNN	off	9	glove	no	89.49

Table 3: Development set performance in percentages.

4.2 Test Set Performance

We selected three models for test set evaluation. The first one is the top performing model on the development set, a CapsNet with no decoder, window size 11, retrainable GloVe embedding and additional word features. The second model is a variant of the first with a decoder. The third model is a CNN variant of the first with no decoder option. Table 4 shows that the first model performed best with a precision of 87.59%, recall of 87.33% and F_1 rate of 87.46%. However, the best CapsNet model only lead its CNN variant by 1.36% while the state-of-the-art model leads the best CapsNet model by 4.16%. A more in-depth analysis is necessary to investigate if the marginal performance difference between CapsNet and CNN models is meaningful in terms of language understanding.

Model	Precision	Recall	$F_{\beta=1}$
Chiu (2016)	91.39	91.85	91.62
CapsNet	87.59	87.33	87.46
CapsNet+decoder	86.40	87.17	86.78
CNN Baseline	85.93	86.23	86.08
CoNLL-2003 Baseline	71.91	50.90	59.61

Table 4: Test set performance in percentages.

5 Error Analysis

On the test data, we make detailed comparisons of our CapsNet and baseline CNN models through key performance metrics, language examples of erroneous predictions and target proximities from reconstructed embeddings.

5.1 Predictions Overview

The confusion matrix in Table 5 shows that our CapsNet performs better than the baseline CNN on most named entity classes, with the exception of the organization classes B-ORG and I-ORG,

[6]This Github repository provided us the source code to implement primary capsule layers and wiring to a decoder

[7]An earlier series of experiments show that re-trainable GloVe embeddings always performed better than non-trainable GloVe embeddings on this dataset.

Entity Class	O	B-LOC	I-LOC	B-PER	I-PER	B-ORG	I-ORG	B-MISC	I-MISC
O	**99.2(99.0)**	00.1(00.1)	00.0(00.0)	00.1(00.2)	00.0(00.1)	00.1(00.2)	00.1(00.2)	00.2(00.1)	00.1(00.1)
B-LOC	00.8(01.0)	**92.5(90.5)**	00.2(00.2)	02.2(01.7)	00.1(00.1)	02.5(04.8)	00.1(00.1)	01.4(01.6)	00.1(00.1)
I-LOC	05.5(03.9)	02.3(01.2)	**86.8(80.2)**	00.4(00.0)	01.2(05.5)	00.0(00.0)	02.3(07.8)	00.0(00.4)	01.6(01.2)
B-PER	01.8(03.7)	01.4(01.6)	00.1(00.1)	**94.6(89.9)**	00.4(00.7)	01.3(03.7)	00.1(00.1)	00.4(00.3)	00.1(00.1)
I-PER	02.2(04.8)	00.0(00.0)	00.1(00.0)	00.9(00.9)	**94.8(92.0)**	00.1(00.0)	02.0(02.3)	00.0(00.0)	00.0(00.1)
B-ORG	04.6(04.2)	05.8(05.1)	00.0(00.0)	06.2(04.0)	00.0(00.0)	**80.3(84.0)**	00.8(01.6)	02.4(01.1)	00.0(00.1)
I-ORG	07.3(05.3)	00.6(00.1)	05.2(04.1)	00.5(00.2)	02.2(02.3)	02.3(02.8)	**79.6(84.1)**	00.4(00.2)	01.2(01.0)
B-MISC	08.8(00.9)	03.4(03.1)	00.0(00.0)	04.3(04.1)	00.0(00.0)	04.3(08.1)	00.0(00.1)	**78.4(74.1)**	00.9(00.0)
I-MISC	18.1(14.4)	00.0(00.0)	01.4(02.3)	00.9(00.9)	01.4(03.2)	00.5(00.9)	06.9(07.4)	03.7(00.4)	**67.1(67.1)**

Table 5: Test Set Confusion Matrix of CapsNet(CNN) percentages of gold label counts. Rows correspond to gold labels and columns correspond to predictions. O refers to not an entity; LOC refers to location; PER refers to person; ORG refers to organization; MISC refers to miscellaneous; B prefix refers to the beginning word of an entity; I prefix refers to the inside of an entity.

where the baseline CNN performs noticeably better. The two models tie on class I-MISC, on which both models perform poorly and predicted 18.1% and 14.4% of the examples to be class O.

5.2 Error Analysis Scope

We investigate the errors through categories and rank the worst examples by descending order of KL-divergence. These categories include overall, per NER gold label, hallucinations [8], missed NER labels [9] and NER label mismatches [10]. For each note-worthy error, we study both CapsNet and CNN predictions. In addition, we compare the reconstructed embeddings against the input target word embeddings.

5.2.1 Common Weaknesses

Many of the worst by KL-divergence examples have the gold label B-MISC or I-MISC. Upon close examination, there are several types of errors which both the CapsNet and baseline CNN models make. The examples are printed in Table 6. Example 1 and 2 show that the models tend to miss semantic nuances. Example 3 and 4 show that the models tend to miss the MISC labels if little information is provided by the canonicalized tokens in the window. Example 5 shows that the models are limited by information available within the input window and thus failed to capture that ”ACCESS” is a heating oil supplier which was hinted in an earlier part of the same sentence. These common weaknesses have to do with the fact that window models are limited to processing information within their window sizes, long-range dependencies are simply out of reach.

[8]When the gold NER label shows no NER ”O” class but a trained model predicts a NER non-”O” class.

[9]When the gold NER label corresponds to a NER non-”O” class but a trained model predicts no NER ”O” class.

[10]When both the gold label and trained model predicts a NER non-”O” class but there is a NER class mismatch.

5.2.2 Common Strengths

From the erroneous predictions, we also saw some common strengths of the two models. Table 7 shows five examples in which the target word is an out-of-vocabulary word. Their gold labels are somewhat debatable and both models managed to make reasonable predictions by the surrounding words in the window. This common ability to fill in the gap has to do with the baseline CNN's ability to summarize N-gram level semantics with convolutions, and the CapsNet's ability to infer part-to-whole relationships with dynamic routing.

5.2.3 Capsule Routing Versus Max-pooling

Many of the errors made by the baseline CNN, but not the CapsNet, come from the gold classes PER, LOC and O. Some examples are printed in Table 8. In examples 11 to 13 the baseline CNN misses the PER classes when the target word is next to a comma or an out-of-vocabulary word. In examples 14 and 15 the model carelessly extended the tail of an organization as an erroneous I-ORG. In examples 16 and 17 the model disregarded the adjective tag JJ of the target word. Overall, the CNN baseline more often dismisses a PER class as an O class[11]. It also hallucinates more than the CapsNet model[12]. We attribute these errors to the over-compression of local, token level information by max-pooling into scalars. They can be avoided by retaining such information in longer capsule vectors.

On the other hand, many of the errors made by the CapsNet but not the baseline CNN come from the gold class I-ORG[13]. Some examples are printed in Table 9. The CapsNet ended the multi-word organization phrase prematurely as O in examples 18 and 19, and as I-LOC prematurely in

[11]The PER as O errors constitute 7.62% of all the baseline CNN errors, but only 4.08% of the CapsNet errors.

[12]The O as non-”O” errors constitute 25.33% of all the CNN errors and only 22.36% of the CapsNet errors.

	Features	Gold	Pred.	Canonicalized Text Window
1	**nymex**, NNP, allCaps	B-MISC	O	floor session gains in light nymex access trade thursday **,** as
2	**sheffield**, NNP, allCaps	B-MISC	B-ORG	first day of the four-day sheffield shield match between tasmania and
3	**of**, IN, lowercase	I-MISC	O	4<s> princess of <unk>(Loine) DGDG/DGDG/DGDG DGDGDGDGDG philippines </s>
4	**conservative**, JJ, upperInitial	B-MISC	O	4<s> <unk>(Pro-European) conservative <unk>(MP) <unk>(Edwina) <unk>(Currie) told the
5	**access**, NNP, allCaps	B-MISC	O	the first few hours of access **.** 4< /s>

Table 6: Test set examples of common errors made by CapsNet and CNN models. n<symbol> refers to n number of adjacent symbols in the text window.

	Features	Gold	Pred.	Canonicalized Text Window
6	**<unk>(Fe)**, NNP, upperInitial	I-ORG	I-LOC	deal between santa <unk>(Fe) would be a **"** <unk>
7	**<unk>(Tia)**, NNP, upperInitial	I-MISC	I-PER	uae **,** venezuela 's <unk>(Tia) <unk>(Juana) and mexico 's <unk>(Isthmus) **.**
8	**<unk>(Herri)**, NNP, upperInitial	I-ORG	I-PER	the protest organized by <unk>(Herri) <unk>(Batasuna) **,** the political wing of
9	**<unk> (Babri)**, NNP, upperInitial	O	B-LOC	proposal in <unk>(remembrance) of the <unk>(Babri) mosque **,** was <unk>(razed)
10	**<unk>(Uzbek)**, NNP, upperInitial	I-MISC	B-ORG	until the 78th minute when <unk>(Uzbek) striker <unk>(Igor) <unk>(Shkvyrin) took advantage

Table 7: Test set examples showing common strengths of CapsNet and CNN models.

	Features	Gold	Pred.	Canonicalized Text Window with Adjacent Predictions
11	**tom/ian**, NNP, upperInitial	B-PER	O	bevan **,** stuart law **,** tom**O** moody**I-PER ,** ian healy
12	**<unk>(Curtly)**, NNP, upperInitial	B-PER	O	<unk> **,** kenneth benjamin **,** <unk>(Curtly)**O** ambrose**I-PER ,** courtney walsh
13	**<unk>(Del)**, NNP, upperInitial	I-PER	O	**,** francesco <unk> **,** alessandro**B-PER** <unk>(Del)**O ,** <unk> <unk> **,** andrea
14	**fans**, NNP, allCaps	O	I-ORG	gunmen wound two manchester**B-ORG** united**I-ORG** fans**I-ORG** in austria **.** 2< /s>
15	**., .**, noinfo	O	I-ORG	<unk>**B-ORG** trust**I-ORG** &**I-ORG** banking**I-ORG** co**I-ORG . I-ORG** 5< /s>
16	**<unk>(65-80-litre)**, JJ, noinfo	O	B-MISC	2<s> he said the <unk>(65-80-litre)**B-MISC** level**O** can be reached in
17	**<unk>(rush-hour)**, JJ, noinfo	O	B-MISC	injured DGDG others in the <unk>(rush-hour)**B-MISC** train**O .** 3< /s>

Table 8: Test set examples showing errors made by the baseline CNN but not the CapsNet. n<symbol> refers to n number of adjacent symbols in the text window.

	Features	Gold	Pred.	Canonicalized Text Window with Adjacent Predictions
18	**army**, NNP, upperInitial	I-ORG	O	offences after three irish**B-MISC** republican**I-MISC** army**O** <unk>(mortar) bombs were found
19	**trading**, NNP, upperInitial	I-ORG	O	to the office**B-ORG** of**I-ORG** fair**I-ORG** trading**O** by <unk> DGDG **,** DGDGDG
20	**heavy**, NNP, upperInitial	I-ORG	I-LOC	4<s> <unk>(Delphis)**B-ORG** hanover**I-LOC** weekly**O** muni bond yields calculated
21	**galati**, NNP, upperInitial	I-ORG	I-LOC	striker <unk> ion of otelul**B-LOC** galati**I-LOC** and defender <unk> <unk> of
22	**spring**, NNP, upperInitial	I-PER	O	from deputy prime minister dick**B-PER** spring**O** who said the honour had

Table 9: Test set examples showing errors made by the CapsNet but not the CNN. n<symbol> refers to n number of adjacent symbols in the text window.

examples 20 and 21. It also ended the person phrase prematurely as O in example 22. We attribute these errors to the over sensitivity of capsules towards boundaries of named entities specifically in our implementation, which seems to have out-weighed the benefits of dynamic routing to recognize part-to-whole relationships. It is possible that with a larger dataset and more routing iterations, better results can be achieved.

5.2.4 Decoder Analysis

When using the trained CapsNet with decoder to make predictions, reconstructed embeddings are generated as by-products. To study the effectiveness of the decoder as a regularizer, we extract the

[13]The I-ORG as non-"I-LOC" errors constitute 10.0% of all the CapsNet errors and only 12.8% of the CNN errors.

	Target	Gold	Pred.	Loss	Nearest neighbors	Canonicalized Text Window
1	**DGDG**	O	O	0.079	province, language, central, maintained, southern	DG, DG, DG, DGDG, DGDG, DGDG, 5< /s>
2	**results**	O	O	0.899	also, part as while s	2<s> <unk> cup <unk> results **.** 4< /s>
3	**hapoel**	B-ORG	B-ORG	0.896	also, part, while, made, as	5<s> hapoel jerusalem DG maccabi tel aviv
4	**suspended**	O	O	0.773	with, as, while, came, also	hockey **-** <unk> <unk> <unk> suspended for one game **.** < /s>
5	**-**	O	O	0.675	came, part, where as while	DG **-** <unk> <unk> <unk> **;** DG
6	**,**	O	O	0.672	<unk>, lisabeth, matesc, assisted-suicide, berlanda	his injuries on thursday night **,** two days after the blast
7	**killed**	O	O	0.670	<unk>, matescu, lisabeth, grossard, miilion	passengers of the van were killed **.** 4< /s>
8	**security**	O	O	-0.99	<unk>, matescu, lisabeth, grossard, dirkx	5<s> security forces said the bombs **,** may
9	**accident**	B-ORG	B-LOC	-0.99	<unk>, matescu, lisabeth, dirkx, grossard	3<s> the <unk> accident occured in <unk> on thursday
10	**clinton**	B-PER	B-PER	0.253	unimported, chindria, orii, wertpapier, pedestrian	4<s> president clinton aims to hold more news
11	**diplomats**	B-PER	B-PER	-0.52	unimported, chindria, orii, wertpapier, pedestrian	4<s> diplomats said general joseph <unk> of
12	**suharto**	B-PER	B-PER	0.253	unimported, chindria, orii, wertpapier, pedestrian	<unk> group run by president suharto 's <unk> son **,** <unk>
13	**all**	O	O	0.249	unimported, chindria, orii, wertpapier, pedestrian	eyptian president hosni mubarak might all meet on saturday to try
14	**with**	O	O	-0.04	melli, jomhuri, po, pao, cluj	out a tax payment schedule with authorities **,** after regional tax
15	**have**	O	O	-0.04	melli, jomhuri, orii, cluj, yvegeny	multinational force would probably not have to make food <unk> or
16	**me**	O	O	0.113	melli, jomhuri, cluj, portuguesa, universitatea	of commons or as placing me under any obligation to vote
17	**european**	B-MISC	B-MISC	0.157	orii, melli, jomhuri, construtorol, gimnasiala	reports of a rift over european policy **,** a spokesman for

Table 10: Examples of the 5 nearest neighbors of reconstructed embeddings. "Loss" refers to cosine proximity loss and n<symbol> refers to n number of such symbols are adjacent in the text window. Notice that although the nearest neighbors of reconstructed embeddings are not relevant to the target word

nearest word neighbors from these reconstructed embeddings to compare with the input target word embeddings. We apply cosine similarity as a distance metric in the retrained GloVe embedding space. Table 10 reveals that the reconstructed embeddings tend to correspond to the availability of token information and new topics, rather than the target word embeddings themselves. Notice that there are a few major groups of nearest neighbors shared across examples, suggesting the model is projecting the reconstructed embeddings to a few specific regions in vector space. Example 1 refers to five prediction cases in which no information is available besides digit and end-of-sentence tags. Example 2 to 5 have their target words positioned next to a chain of out-of-vocabulary or start-of-sentence tokens. Example 6 to 9 came from news topics about violence. Example 10 to 13 have windows which include the word "president". Examples 14 to 17 came from news topics about governments and politics. Since inputs for the decoder are wired and masked from the final layer capsules in the main model, this observation suggests that the isolated final layer capsules has a good level of window context information encoded. Effectively, CapsNet does not simply retain sensitive entity-level boundary information in Capsules, but also routes whole context level information to them.

6 Concluding Remarks

In this paper, we investigated the viability of Capsule Network for named entity recognition using the CoNLL-2003 English dataset. We adapted the architecture proposed by Sabour et al. to windowed language inputs and named entity class outputs. Experiment results on the training set show that various CapsNets generally perform better than their analogous CNNs of the same depth, and results on the test set show that CapsNet performed marginally better overall, and noticeably better on 6 out of 9 entity classes. Our in-depth analysis with tabulated examples shows that both models suffer from the limitation of window sizes, but demonstrate the ability to make sensible inference for out-of-vocabulary words. However, the models also noticeably differentiate from each other when it comes to detecting boundaries for multi-word named entities. While the baseline CNN often insensitively extends the boundaries to one too many tokens, the CapsNet often prematurely terminates or divides up multi-word entities. The addition of a decoder as a regularizer to the main CapsNet model doesn't seem to improve overall performance due to the additional trainable parameters it requires. We anticipate that with a more extensive exploration on the number of dynamic routing iterations and a larger dataset, CapsNet can be trained to better recognize part-to-whole relationships.

This basic experiment warrants many promising directions for future explorations. Building on other named entity research works, including those cited in the background section, sophisticated pre-trained word level and character level features can be concatenated with the basic embedding inputs; hybrid architecture can be explored in combination with CapsNet; new and larger datasets can be analyzed in parallel. To our best knowledge, this is the first research project which investigated CapsNet against analogous CNN for named entity recognition with an in-depth error analysis. We hope this paper can contribute to laying the foundation for adapting and refining Capsule Networks for natural language processing purposes.

Our implementation is available upon request.

7 Acknowledgements

This research was supported by the Master of Information and Data Science program at UC Berkeley School of Information. We gratefully acknowledge the feedback from the instructors of course W266.

References

Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, 2015.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics, 2003.

James Hammerton. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics, 2003.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

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

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

Guo Xifeng. A keras implementation of capsnet in nips2017 paper "dynamic routing between capsules". `https://github.com/XifengGuo/CapsNet-Keras`, 2017.