Introduction to Deep Learning in NLP

Deep Learning and the Nature of Linguistic Representation

Shalom Lappin

University of Gothenburg, Queen Mary University of London, and King's College London

WeSSLLI 2020, Brandeis University

July 13, 2020

Outline

Schedule of Topics

All times given are EST.

- Class 1: July 13, 11:00-12:20
  Introduction to Deep Learning in NLP
- Class 2: July 14, 11:00-12:20
  Learning Syntactic Properties with Deep Neural Networks
- Class 3: July 15, 11:00-12:20
  Machine Learning and the Sentence Acceptability Task
- Class 4: July 16, 11:00-12:20
  Predicting Human Acceptability Judgments in Context
- Class 5: July 17, 11:00-12:20
  Cognitively Viable Computational Models of Linguistic Knowledge

Schedule of Topics

All times given are EST.

- Class 1: July 13, 11:00-12:20
  Introduction to Deep Learning in NLP
- Class 2: July 14, 11:00-12:20
  Learning Syntactic Properties with Deep Neural Networks
- Class 3: July 15, 11:00-12:20
  Machine Learning and the Sentence Acceptability Task
- Class 4: July 16, 11:00-12:20
  Predicting Human Acceptability Judgments in Context
- Class 5: July 17, 11:00-12:20
  Cognitively Viable Computational Models of Linguistic Knowledge

Schedule of Topics

All times given are EST.

- Class 1: July 13, 11:00-12:20
  Introduction to Deep Learning in NLP
- Class 2: July 14, 11:00-12:20
  Learning Syntactic Properties with Deep Neural Networks
- Class 3: July 15, 11:00-12:20
  Machine Learning and the Sentence Acceptability Task
- Class 4: July 16, 11:00-12:20
  Predicting Human Acceptability Judgments in Context
- Class 5: July 17, 11:00-12:20
  Cognitively Viable Computational Models of Linguistic Knowledge

Schedule of Topics

All times given are EST.

- Class 1: July 13, 11:00-12:20
  Introduction to Deep Learning in NLP
- Class 2: July 14, 11:00-12:20
  Learning Syntactic Properties with Deep Neural Networks
- Class 3: July 15, 11:00-12:20
  Machine Learning and the Sentence Acceptability Task
- Class 4: July 16, 11:00-12:20
  Predicting Human Acceptability Judgments in Context
- Class 5: July 17, 11:00-12:20
  Cognitively Viable Computational Models of Linguistic Knowledge

Schedule of Topics

All times given are EST.

- Class 1: July 13, 11:00-12:20
  Introduction to Deep Learning in NLP
- Class 2: July 14, 11:00-12:20
  Learning Syntactic Properties with Deep Neural Networks
- Class 3: July 15, 11:00-12:20
  Machine Learning and the Sentence Acceptability Task
- Class 4: July 16, 11:00-12:20
  Predicting Human Acceptability Judgments in Context
- Class 5: July 17, 11:00-12:20
  Cognitively Viable Computational Models of Linguistic Knowledge

The Dominance of Deep Learning in AI

- Over the past ten years the emergence of powerful deep learning (DL) models has produced significant advances across a wide range of AI tasks and domains.
- These include, among others, image classification, face recognition, medical diagnostics, game playing, and autonomous robots.
- DL has been particularly influential in NLP, where it has yielded substantial progress in applications like machine translation, speech recognition, question-answering, dialogue management, paraphrase identification, and NL inference.
- In these areas of research it has displaced other machine learning methods to become the dominant approach.

The Dominance of Deep Learning in AI

- Over the past ten years the emergence of powerful deep learning (DL) models has produced significant advances across a wide range of AI tasks and domains.
- These include, among others, image classification, face recognition, medical diagnostics, game playing, and autonomous robots.
- DL has been particularly influential in NLP, where it has yielded substantial progress in applications like machine translation, speech recognition, question-answering, dialogue management, paraphrase identification, and NL inference.
- In these areas of research it has displaced other machine learning methods to become the dominant approach.

The Dominance of Deep Learning in AI

- Over the past ten years the emergence of powerful deep learning (DL) models has produced significant advances across a wide range of AI tasks and domains.
- These include, among others, image classification, face recognition, medical diagnostics, game playing, and autonomous robots.
- DL has been particularly influential in NLP, where it has yielded substantial progress in applications like machine translation, speech recognition, question-answering, dialogue management, paraphrase identification, and NL inference.
- In these areas of research it has displaced other machine learning methods to become the dominant approach.

The Dominance of Deep Learning in AI

- Over the past ten years the emergence of powerful deep learning (DL) models has produced significant advances across a wide range of AI tasks and domains.
- These include, among others, image classification, face recognition, medical diagnostics, game playing, and autonomous robots.
- DL has been particularly influential in NLP, where it has yielded substantial progress in applications like machine translation, speech recognition, question-answering, dialogue management, paraphrase identification, and NL inference.
- In these areas of research it has displaced other machine learning methods to become the dominant approach.

From Engineering to Cognitive Science

- The success of DL as an engineering method raises important cognitive issues.
- Deep Neural Networks (DNNs) constitute domain general learning devices, which apply the same basic approach to learning, data processing, and representation in all domains.
- If they are able to approximate or surpass human performance in a task, what conclusions, if any, can we draw concerning the nature of human learning and representation for that task?

From Engineering to Cognitive Science

- The success of DL as an engineering method raises important cognitive issues.
- Deep Neural Networks (DNNs) constitute domain general learning devices, which apply the same basic approach to learning, data processing, and representation in all domains.
- If they are able to approximate or surpass human performance in a task, what conclusions, if any, can we draw concerning the nature of human learning and representation for that task?

From Engineering to Cognitive Science

- The success of DL as an engineering method raises important cognitive issues.
- Deep Neural Networks (DNNs) constitute domain general learning devices, which apply the same basic approach to learning, data processing, and representation in all domains.
- If they are able to approximate or surpass human performance in a task, what conclusions, if any, can we draw concerning the nature of human learning and representation for that task?

Machine Learning as a Source of Insight into Human Linguistic Knowledge

- Lappin and Shieber (2007), and Clark and Lappin (2011) suggest that computational learning theory provides a guide to determining how much linguistic knowledge can be acquired through different types of machine learning models.
- They argue that relatively weak bias models can efficiently learn complex grammar classes suitable for NL syntax.
- Their results do not entail that humans actually use these models for language acquisition.
- But they do indicate the classes of grammars that humans can, in principle, acquire through domain general methods of induction, from tractable quantities of data, in plausible amounts of time.

Machine Learning as a Source of Insight into Human Linguistic Knowledge

- Lappin and Shieber (2007), and Clark and Lappin (2011) suggest that computational learning theory provides a guide to determining how much linguistic knowledge can be acquired through different types of machine learning models.
- They argue that relatively weak bias models can efficiently learn complex grammar classes suitable for NL syntax.
- Their results do not entail that humans actually use these models for language acquisition.
- But they do indicate the classes of grammars that humans can, in principle, acquire through domain general methods of induction, from tractable quantities of data, in plausible amounts of time.

Machine Learning as a Source of Insight into Human Linguistic Knowledge

- Lappin and Shieber (2007), and Clark and Lappin (2011) suggest that computational learning theory provides a guide to determining how much linguistic knowledge can be acquired through different types of machine learning models.
- They argue that relatively weak bias models can efficiently learn complex grammar classes suitable for NL syntax.
- Their results do not entail that humans actually use these models for language acquisition.
- But they do indicate the classes of grammars that humans can, in principle, acquire through domain general methods of induction, from tractable quantities of data, in plausible amounts of time.

Machine Learning as a Source of Insight into Human Linguistic Knowledge

- Lappin and Shieber (2007), and Clark and Lappin (2011) suggest that computational learning theory provides a guide to determining how much linguistic knowledge can be acquired through different types of machine learning models.
- They argue that relatively weak bias models can efficiently learn complex grammar classes suitable for NL syntax.
- Their results do not entail that humans actually use these models for language acquisition.
- But they do indicate the classes of grammars that humans can, in principle, acquire through domain general methods of induction, from tractable quantities of data, in plausible amounts of time.

DL and Human Cognition

- Early connectionists (Rumelhart, McClelland, and the PDP Research Group, 1986) claimed that NNs are modelled on the human brain.
- Few people working in DL today make this strong claim.
- The extent, if any, to which human learning and representation resemble those of a DNN can only be determined by neuroscientific research.
- In this course we are not focussing on grammar induction, but the more general issues of language learning and the nature of linguistic representation.
- The weaker view applies: If a DNN is able to approach or surpass human performance on a linguistic task, then this shows how domain general learning mechanisms, possibly supplemented with additional domain bias factors, can, in principle, acquire this knowledge efficiently.

DL and Human Cognition

- Early connectionists (Rumelhart, McClelland, and the PDP Research Group, 1986) claimed that NNs are modelled on the human brain.
- Few people working in DL today make this strong claim.
- The extent, if any, to which human learning and representation resemble those of a DNN can only be determined by neuroscientific research.
- In this course we are not focussing on grammar induction, but the more general issues of language learning and the nature of linguistic representation.
- The weaker view applies: If a DNN is able to approach or surpass human performance on a linguistic task, then this shows how domain general learning mechanisms, possibly supplemented with additional domain bias factors, can, in principle, acquire this knowledge efficiently.

DL and Human Cognition

- Early connectionists (Rumelhart, McClelland, and the PDP Research Group, 1986) claimed that NNs are modelled on the human brain.
- Few people working in DL today make this strong claim.
- The extent, if any, to which human learning and representation resemble those of a DNN can only be determined by neuroscientific research.
- In this course we are not focussing on grammar induction, but the more general issues of language learning and the nature of linguistic representation.
- The weaker view applies: If a DNN is able to approach or surpass human performance on a linguistic task, then this shows how domain general learning mechanisms, possibly supplemented with additional domain bias factors, can, in principle, acquire this knowledge efficiently.

DL and Human Cognition

- Early connectionists (Rumelhart, McClelland, and the PDP Research Group, 1986) claimed that NNs are modelled on the human brain.
- Few people working in DL today make this strong claim.
- The extent, if any, to which human learning and representation resemble those of a DNN can only be determined by neuroscientific research.
- In this course we are not focussing on grammar induction, but the more general issues of language learning and the nature of linguistic representation.
- The weaker view applies: If a DNN is able to approach or surpass human performance on a linguistic task, then this shows how domain general learning mechanisms, possibly supplemented with additional domain bias factors, can, in principle, acquire this knowledge efficiently.

DL and Human Cognition

- Early connectionists (Rumelhart, McClelland, and the PDP Research Group, 1986) claimed that NNs are modelled on the human brain.
- Few people working in DL today make this strong claim.
- The extent, if any, to which human learning and representation resemble those of a DNN can only be determined by neuroscientific research.
- In this course we are not focussing on grammar induction, but the more general issues of language learning and the nature of linguistic representation.
- The weaker view applies: If a DNN is able to approach or surpass human performance on a linguistic task, then this shows how domain general learning mechanisms, possibly supplemented with additional domain bias factors, can, in principle, acquire this knowledge efficiently.

Architecture of Deep Feed Forward Networks

- DNNs learn an approximation of a function $f(x) = y$ which maps input data x to an output value y (category assignment, probability distribution, etc.).
- Deep Feed Forward Networks consist of (1) an input layer where data is entered, (2) one or more hidden layers, in which units (neurons) compute the weights for components of the data, and (3) an output layer that generates a value for the function.
- DNNs can, in principle, approximate any function, and, in particular, non-linear functions.
- Sigmoid functions are commonly used to determine the activation threshold for a neuron.

Architecture of Deep Feed Forward Networks

- DNNs learn an approximation of a function $f(x) = y$ which maps input data x to an output value y (category assignment, probability distribution, etc.).
- Deep Feed Forward Networks consist of (1) an input layer where data is entered, (2) one or more hidden layers, in which units (neurons) compute the weights for components of the data, and (3) an output layer that generates a value for the function.
- DNNs can, in principle, approximate any function, and, in particular, non-linear functions.
- Sigmoid functions are commonly used to determine the activation threshold for a neuron.

Architecture of Deep Feed Forward Networks

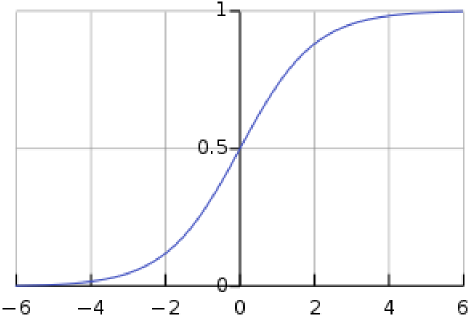
- DNNs learn an approximation of a function $f(x) = y$ which maps input data x to an output value y (category assignment, probability distribution, etc.).
- Deep Feed Forward Networks consist of (1) an input layer where data is entered, (2) one or more hidden layers, in which units (neurons) compute the weights for components of the data, and (3) an output layer that generates a value for the function.
- DNNs can, in principle, approximate any function, and, in particular, non-linear functions.
- Sigmoid functions are commonly used to determine the activation threshold for a neuron.

Architecture of Deep Feed Forward Networks

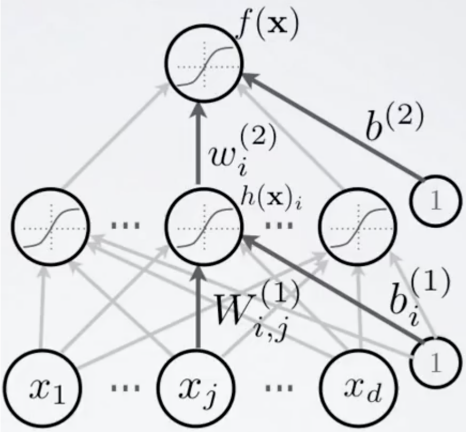
- DNNs learn an approximation of a function $f(x) = y$ which maps input data x to an output value y (category assignment, probability distribution, etc.).
- Deep Feed Forward Networks consist of (1) an input layer where data is entered, (2) one or more hidden layers, in which units (neurons) compute the weights for components of the data, and (3) an output layer that generates a value for the function.
- DNNs can, in principle, approximate any function, and, in particular, non-linear functions.
- Sigmoid functions are commonly used to determine the activation threshold for a neuron.

Example of a Sigmoid Function: Logistic Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Deep Feed Forward Networks



From Tushar Gupta, "Deep Learning: Feedforward Neural Network", *Towards Data Science*, Jan 5, 2017

Training a DNN

- Training a DNN involves comparing its predicted function value to the ground truth of training data.
- Its error rate is reduced in cycles (epochs) through back propagation.
- This process involves computing the gradient of a loss (error) function, and proceeding down the slope, by specified increments, to an estimated optimal level, determined by stochastic gradient descent.

Training a DNN

- Training a DNN involves comparing its predicted function value to the ground truth of training data.
- Its error rate is reduced in cycles (epochs) through back propagation.
- This process involves computing the gradient of a loss (error) function, and proceeding down the slope, by specified increments, to an estimated optimal level, determined by stochastic gradient descent.

Training a DNN

- Training a DNN involves comparing its predicted function value to the ground truth of training data.
- Its error rate is reduced in cycles (epochs) through back propagation.
- This process involves computing the gradient of a loss (error) function, and proceeding down the slope, by specified increments, to an estimated optimal level, determined by stochastic gradient descent.

Using Cross Entropy as an Error Function

- Cross Entropy is a function that measures the difference between two probability distributions P and Q: $H(P, Q) = -E_{x \sim P} \, log \, Q(x)$
- It is widely used as a loss function for gradient descent in training DNNs.
- At each epoch in the training process cross entropy is computed, and the values of the weights assigned to the hidden units are adjusted to reduce error along the slope identified by gradient descent.
- Training is concluded when the distance between the network's predicted distribution, and that projected from the training data, reach an estimated optimal minimum.

Using Cross Entropy as an Error Function

- Cross Entropy is a function that measures the difference between two probability distributions P and Q: $H(P, Q) = -E_{x \sim P} \, log \, Q(x)$
- It is widely used as a loss function for gradient descent in training DNNs.
- At each epoch in the training process cross entropy is computed, and the values of the weights assigned to the hidden units are adjusted to reduce error along the slope identified by gradient descent.
- Training is concluded when the distance between the network's predicted distribution, and that projected from the training data, reach an estimated optimal minimum.

Using Cross Entropy as an Error Function

- Cross Entropy is a function that measures the difference between two probability distributions P and Q: $H(P, Q) = -E_{x \sim P}\ log\ Q(x)$
- It is widely used as a loss function for gradient descent in training DNNs.
- At each epoch in the training process cross entropy is computed, and the values of the weights assigned to the hidden units are adjusted to reduce error along the slope identified by gradient descent.
- Training is concluded when the distance between the network's predicted distribution, and that projected from the training data, reach an estimated optimal minimum.

Using Cross Entropy as an Error Function

- Cross Entropy is a function that measures the difference between two probability distributions P and Q: $H(P, Q) = -E_{x \sim P}\, log\ Q(x)$
- It is widely used as a loss function for gradient descent in training DNNs.
- At each epoch in the training process cross entropy is computed, and the values of the weights assigned to the hidden units are adjusted to reduce error along the slope identified by gradient descent.
- Training is concluded when the distance between the network's predicted distribution, and that projected from the training data, reach an estimated optimal minimum.

Generating a Probability Distribution over Random Variables with Softmax

- In many cases the hidden layers of a DNN will produce a set of non-normalised probability scores for the different states of a random variable corresponding to a category judgment, or the likelihood of an event.
- The softmax function maps the vector of these scores into a normalised probability distribution whose values sum to 1.
- $softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$
- The softmax function is widely used in the output layer of a DNN to generate a probability distribution for a classifier, or for a probability model.

Generating a Probability Distribution over Random Variables with Softmax

- In many cases the hidden layers of a DNN will produce a set of non-normalised probability scores for the different states of a random variable corresponding to a category judgment, or the likelihood of an event.
- The softmax function maps the vector of these scores into a normalised probability distribution whose values sum to 1.
- $softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$
- The softmax function is widely used in the output layer of a DNN to generate a probability distribution for a classifier, or for a probability model.

Generating a Probability Distribution over Random Variables with Softmax

- In many cases the hidden layers of a DNN will produce a set of non-normalised probability scores for the different states of a random variable corresponding to a category judgment, or the likelihood of an event.
- The softmax function maps the vector of these scores into a normalised probability distribution whose values sum to 1.
- $softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$
- The softmax function is widely used in the output layer of a DNN to generate a probability distribution for a classifier, or for a probability model.

Generating a Probability Distribution over Random Variables with Softmax

- In many cases the hidden layers of a DNN will produce a set of non-normalised probability scores for the different states of a random variable corresponding to a category judgment, or the likelihood of an event.
- The softmax function maps the vector of these scores into a normalised probability distribution whose values sum to 1.
- $softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$
- The softmax function is widely used in the output layer of a DNN to generate a probability distribution for a classifier, or for a probability model.

Word Embeddings

- Words are represented in a DNN by vectors of real numbers.
- Each element of the vector expresses a distributional feature of the word.
- These features are the dimensions of the vectors, and they encode its co-occurrence patterns with other words in a training corpus.
- Word embeddings are generally compressed into low dimensional vectors (200-300 dimensions) that express similarity and proximity relations among the words in the vocabulary of a DNN model.
- These models frequently use large pre-trained word embeddings, like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), compiled from millions of words of text.

Word Embeddings

- Words are represented in a DNN by vectors of real numbers.
- Each element of the vector expresses a distributional feature of the word.
- These features are the dimensions of the vectors, and they encode its co-occurrence patterns with other words in a training corpus.
- Word embeddings are generally compressed into low dimensional vectors (200-300 dimensions) that express similarity and proximity relations among the words in the vocabulary of a DNN model.
- These models frequently use large pre-trained word embeddings, like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), compiled from millions of words of text.

Word Embeddings

- Words are represented in a DNN by vectors of real numbers.
- Each element of the vector expresses a distributional feature of the word.
- These features are the dimensions of the vectors, and they encode its co-occurrence patterns with other words in a training corpus.
- Word embeddings are generally compressed into low dimensional vectors (200-300 dimensions) that express similarity and proximity relations among the words in the vocabulary of a DNN model.
- These models frequently use large pre-trained word embeddings, like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), compiled from millions of words of text.

Word Embeddings

- Words are represented in a DNN by vectors of real numbers.
- Each element of the vector expresses a distributional feature of the word.
- These features are the dimensions of the vectors, and they encode its co-occurrence patterns with other words in a training corpus.
- Word embeddings are generally compressed into low dimensional vectors (200-300 dimensions) that express similarity and proximity relations among the words in the vocabulary of a DNN model.
- These models frequently use large pre-trained word embeddings, like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), compiled from millions of words of text.

Word Embeddings

- Words are represented in a DNN by vectors of real numbers.
- Each element of the vector expresses a distributional feature of the word.
- These features are the dimensions of the vectors, and they encode its co-occurrence patterns with other words in a training corpus.
- Word embeddings are generally compressed into low dimensional vectors (200-300 dimensions) that express similarity and proximity relations among the words in the vocabulary of a DNN model.
- These models frequently use large pre-trained word embeddings, like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), compiled from millions of words of text.

Two Types of Learning

- In supervised learning a DNN is trained on data annotated with the features that it is learning to predict.
- For example, if the DNN is learning to identify the objects that appear in graphic images, then its training data may consist of large numbers of labelled images of the objects that it is intended to recognise in photographs.
- In unsupervised learning the training data is not labelled.
- A generative neural language model may be trained on large quantities of raw text.
- It will generate the most likely word in a sequence, given the previous words, on the basis of the probability distribution over words, and sequences of words, that it estimates from the unlabelled training corpus.

Two Types of Learning

- In supervised learning a DNN is trained on data annotated with the features that it is learning to predict.
- For example, if the DNN is learning to identify the objects that appear in graphic images, then its training data may consist of large numbers of labelled images of the objects that it is intended to recognise in photographs.
- In unsupervised learning the training data is not labelled.
- A generative neural language model may be trained on large quantities of raw text.
- It will generate the most likely word in a sequence, given the previous words, on the basis of the probability distribution over words, and sequences of words, that it estimates from the unlabelled training corpus.

Two Types of Learning

- In supervised learning a DNN is trained on data annotated with the features that it is learning to predict.
- For example, if the DNN is learning to identify the objects that appear in graphic images, then its training data may consist of large numbers of labelled images of the objects that it is intended to recognise in photographs.
- In unsupervised learning the training data is not labelled.
- A generative neural language model may be trained on large quantities of raw text.
- It will generate the most likely word in a sequence, given the previous words, on the basis of the probability distribution over words, and sequences of words, that it estimates from the unlabelled training corpus.

Two Types of Learning

- In supervised learning a DNN is trained on data annotated with the features that it is learning to predict.
- For example, if the DNN is learning to identify the objects that appear in graphic images, then its training data may consist of large numbers of labelled images of the objects that it is intended to recognise in photographs.
- In unsupervised learning the training data is not labelled.
- A generative neural language model may be trained on large quantities of raw text.
- It will generate the most likely word in a sequence, given the previous words, on the basis of the probability distribution over words, and sequences of words, that it estimates from the unlabelled training corpus.

Two Types of Learning

- In supervised learning a DNN is trained on data annotated with the features that it is learning to predict.
- For example, if the DNN is learning to identify the objects that appear in graphic images, then its training data may consist of large numbers of labelled images of the objects that it is intended to recognise in photographs.
- In unsupervised learning the training data is not labelled.
- A generative neural language model may be trained on large quantities of raw text.
- It will generate the most likely word in a sequence, given the previous words, on the basis of the probability distribution over words, and sequences of words, that it estimates from the unlabelled training corpus.

Recurrent Neural Networks

- Feed Forward Neural Networks take data encoded in vectors of fixed size as input, and they yield output vectors of fixed size.
- Recurrent Neural Networks (RNNs, Elman, 1990) apply to sequences of input vectors, producing a string of output vectors.
- They retain information from previous processing phases in a sequence, and so they have a memory over the span of the input.
- RNNs are particularly well suited to processing natural language, whose units of sound and text are structured as ordered strings.

Recurrent Neural Networks

- Feed Forward Neural Networks take data encoded in vectors of fixed size as input, and they yield output vectors of fixed size.
- Recurrent Neural Networks (RNNs, Elman, 1990) apply to sequences of input vectors, producing a string of output vectors.
- They retain information from previous processing phases in a sequence, and so they have a memory over the span of the input.
- RNNs are particularly well suited to processing natural language, whose units of sound and text are structured as ordered strings.

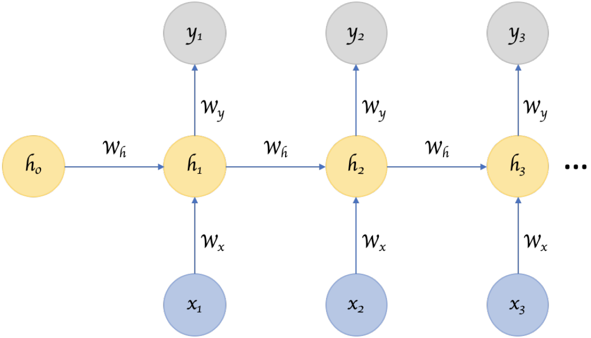
Recurrent Neural Networks

- Feed Forward Neural Networks take data encoded in vectors of fixed size as input, and they yield output vectors of fixed size.
- Recurrent Neural Networks (RNNs, Elman, 1990) apply to sequences of input vectors, producing a string of output vectors.
- They retain information from previous processing phases in a sequence, and so they have a memory over the span of the input.
- RNNs are particularly well suited to processing natural language, whose units of sound and text are structured as ordered strings.

Recurrent Neural Networks

- Feed Forward Neural Networks take data encoded in vectors of fixed size as input, and they yield output vectors of fixed size.
- Recurrent Neural Networks (RNNs, Elman, 1990) apply to sequences of input vectors, producing a string of output vectors.
- They retain information from previous processing phases in a sequence, and so they have a memory over the span of the input.
- RNNs are particularly well suited to processing natural language, whose units of sound and text are structured as ordered strings.

Structure of an RNN



From Mahendran Venkatachalam, "Recurrent Neural Networks: Remembering What's Important", *Towards Data Science*, March 1, 2019

LSTMs

- Simple RNNs preserve information from previous states, but they do not effectively control this information.
- They have difficulties representing long distance dependencies between elements of a sequence.
- An LSTM (Hochreiter and Schmidhuber, 1997) is a type of RNN whose units contain three types of information gates, composed of sigmoid and tanh functions.
- (i) The forgetting gate determines which part of the information received from preceding units is discarded; (ii) the input gate updates the retained information with the features of a new element of the input sequence; and (iii) the output gate defines the vector which is passed to the next unit in the network.

LSTMs

- Simple RNNs preserve information from previous states, but they do not effectively control this information.
- They have difficulties representing long distance dependencies between elements of a sequence.
- An LSTM (Hochreiter and Schmidhuber, 1997) is a type of RNN whose units contain three types of information gates, composed of sigmoid and tanh functions.
- (i) The forgetting gate determines which part of the information received from preceding units is discarded; (ii) the input gate updates the retained information with the features of a new element of the input sequence; and (iii) the output gate defines the vector which is passed to the next unit in the network.

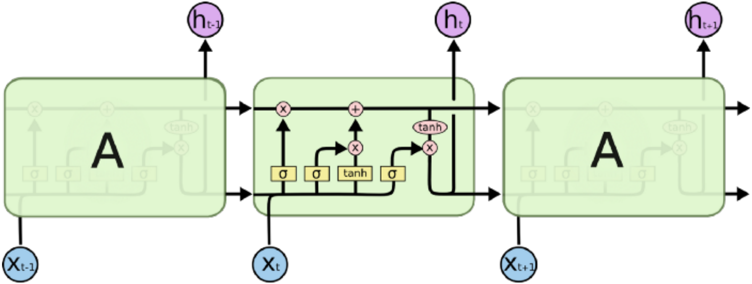
LSTMs

- Simple RNNs preserve information from previous states, but they do not effectively control this information.
- They have difficulties representing long distance dependencies between elements of a sequence.
- An LSTM (Hochreiter and Schmidhuber, 1997) is a type of RNN whose units contain three types of information gates, composed of sigmoid and tanh functions.
- (i) The forgetting gate determines which part of the information received from preceding units is discarded; (ii) the input gate updates the retained information with the features of a new element of the input sequence; and (iii) the output gate defines the vector which is passed to the next unit in the network.

LSTMs

- Simple RNNs preserve information from previous states, but they do not effectively control this information.
- They have difficulties representing long distance dependencies between elements of a sequence.
- An LSTM (Hochreiter and Schmidhuber, 1997) is a type of RNN whose units contain three types of information gates, composed of sigmoid and tanh functions.
- (i) The forgetting gate determines which part of the information received from preceding units is discarded; (ii) the input gate updates the retained information with the features of a new element of the input sequence; and (iii) the output gate defines the vector which is passed to the next unit in the network.

LSTM Architecture



From Christopher Olah's blog *Understanding LSTM Networks*, August 27, 2015

Convolutional Neural Networks

- In a convolutional neural network (CNN) input is fed to a convolutional layer, which extracts a feature map from this data.
- A pooling layer compresses the map by reducing its dimensions, and rendering it invariant to small changes in input (noise filtering).
- Successive convolutional + pooling layers construct progressively higher level representations from the feature maps received from preceding levels of the network.
- The output feature map is passed to one or more fully interconnected layers, which transform the map into a feature vector.
- A softmax function maps this vector into a probability distribution over the the states of a category variable.

Convolutional Neural Networks

- In a convolutional neural network (CNN) input is fed to a convolutional layer, which extracts a feature map from this data.
- A pooling layer compresses the map by reducing its dimensions, and rendering it invariant to small changes in input (noise filtering).
- Successive convolutional + pooling layers construct progressively higher level representations from the feature maps received from preceding levels of the network.
- The output feature map is passed to one or more fully interconnected layers, which transform the map into a feature vector.
- A softmax function maps this vector into a probability distribution over the the states of a category variable.

Convolutional Neural Networks

- In a convolutional neural network (CNN) input is fed to a convolutional layer, which extracts a feature map from this data.
- A pooling layer compresses the map by reducing its dimensions, and rendering it invariant to small changes in input (noise filtering).
- Successive convolutional + pooling layers construct progressively higher level representations from the feature maps received from preceding levels of the network.
- The output feature map is passed to one or more fully interconnected layers, which transform the map into a feature vector.
- A softmax function maps this vector into a probability distribution over the the states of a category variable.

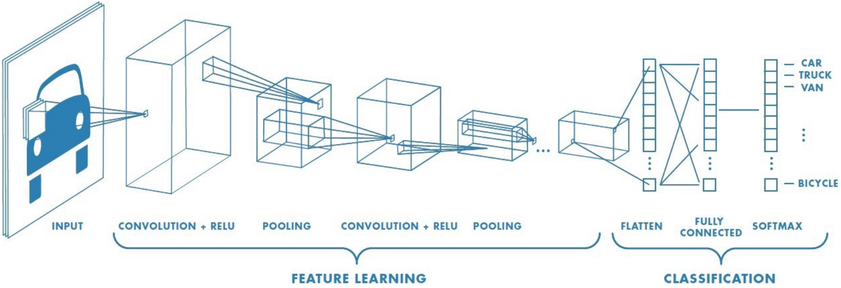
Convolutional Neural Networks

- In a convolutional neural network (CNN) input is fed to a convolutional layer, which extracts a feature map from this data.
- A pooling layer compresses the map by reducing its dimensions, and rendering it invariant to small changes in input (noise filtering).
- Successive convolutional + pooling layers construct progressively higher level representations from the feature maps received from preceding levels of the network.
- The output feature map is passed to one or more fully interconnected layers, which transform the map into a feature vector.
- A softmax function maps this vector into a probability distribution over the the states of a category variable.

Convolutional Neural Networks

- In a convolutional neural network (CNN) input is fed to a convolutional layer, which extracts a feature map from this data.
- A pooling layer compresses the map by reducing its dimensions, and rendering it invariant to small changes in input (noise filtering).
- Successive convolutional + pooling layers construct progressively higher level representations from the feature maps received from preceding levels of the network.
- The output feature map is passed to one or more fully interconnected layers, which transform the map into a feature vector.
- A softmax function maps this vector into a probability distribution over the the states of a category variable.

Example of a CNN



From Sumit Saha "A Comprehensive Guide to Convolutional Neural Networks– the ELI5 Way", *Towards Data Science,* December 15, 2018

Attention

- Attention was developed to solve a problem in seq2seq neural machine translation, which uses an encoder-decoder architecture.
- In earlier versions of this architecture an RNN (or LSTM) encoded an input sequence as a single context vector, which a decoder RNN (LSTM) mapped to a target language sequence.
- Information from the previous hidden states of the encoder is lost, and all the words in the encoder's output vector are given equal weight when it is passed to the decoder.

Attention

- Attention was developed to solve a problem in seq2seq neural machine translation, which uses an encoder-decoder architecture.
- In earlier versions of this architecture an RNN (or LSTM) encoded an input sequence as a single context vector, which a decoder RNN (LSTM) mapped to a target language sequence.
- Information from the previous hidden states of the encoder is lost, and all the words in the encoder's output vector are given equal weight when it is passed to the decoder.

Attention

- Attention was developed to solve a problem in seq2seq neural machine translation, which uses an encoder-decoder architecture.
- In earlier versions of this architecture an RNN (or LSTM) encoded an input sequence as a single context vector, which a decoder RNN (LSTM) mapped to a target language sequence.
- Information from the previous hidden states of the encoder is lost, and all the words in the encoder's output vector are given equal weight when it is passed to the decoder.

Attention and Self-Attention

- Bahdanau et al. (2015) introduce an attention layer that computes relative weights for each of the words in the input sequence, and these are combined with the context vector.
- The attention mechanism significantly improves the accuracy of seq2seq word alignment.
- It learns the relative importance of elements in the input in determining correspondences to elements in the output sequence.
- Self-attention identifies relations among the elements of the same sequence, which enhances the capacity of the DNN to recognise long distance dependency patterns in that sequence.

Attention and Self-Attention

- Bahdanau et al. (2015) introduce an attention layer that computes relative weights for each of the words in the input sequence, and these are combined with the context vector.
- The attention mechanism significantly improves the accuracy of seq2seq word alignment.
- It learns the relative importance of elements in the input in determining correspondences to elements in the output sequence.
- Self-attention identifies relations among the elements of the same sequence, which enhances the capacity of the DNN to recognise long distance dependency patterns in that sequence.

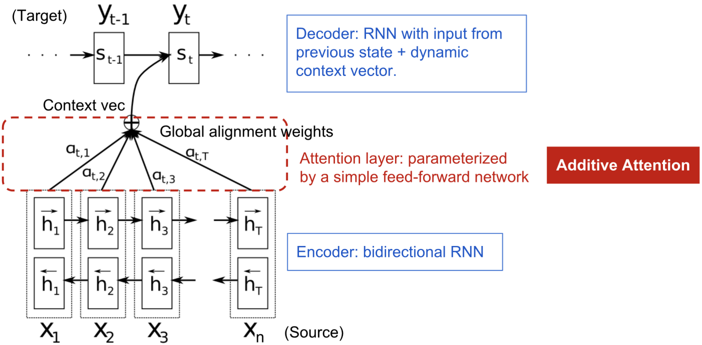
Attention and Self-Attention

- Bahdanau et al. (2015) introduce an attention layer that computes relative weights for each of the words in the input sequence, and these are combined with the context vector.
- The attention mechanism significantly improves the accuracy of seq2seq word alignment.
- It learns the relative importance of elements in the input in determining correspondences to elements in the output sequence.
- Self-attention identifies relations among the elements of the same sequence, which enhances the capacity of the DNN to recognise long distance dependency patterns in that sequence.

Attention and Self-Attention

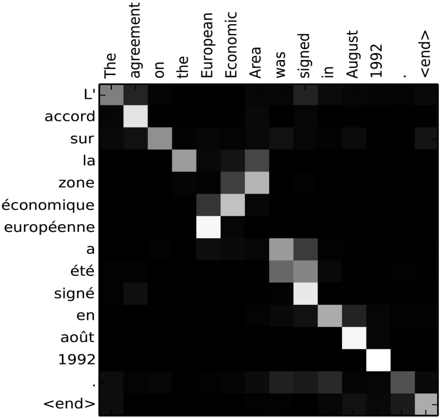
- Bahdanau et al. (2015) introduce an attention layer that computes relative weights for each of the words in the input sequence, and these are combined with the context vector.
- The attention mechanism significantly improves the accuracy of seq2seq word alignment.
- It learns the relative importance of elements in the input in determining correspondences to elements in the output sequence.
- Self-attention identifies relations among the elements of the same sequence, which enhances the capacity of the DNN to recognise long distance dependency patterns in that sequence.

Attention In Neural Machine Translation



From Dzmitry Bahdanau et al. (2015), "Neural Machine Translation by Jointly Learning to Align and Translate", *ICLR 2015*

Word Alignment with Attention Weights



From Dzmitry Bahdanau et al. (2015), "Neural Machine Translation by Jointly Learning to Align and Translate", *ICLR 2015*

Transformers

- Transformers (Vaswani et al., 2017) dispense with recurrent networks and convolution.
- Instead they construct both encoders and decoders out of stacks of layers that consist of multi-head attention units which provide input to a feed forward network.
- These layers process input sequences simultaneously, in parallel, without considering sequential order.
- The relative positions of the elements of a sequence are represented as additional information.
- The attention driven design of transformers has allowed them to achieve a significant improvement over LSTMs and CNNs, across a wide range of tasks.

Transformers

- Transformers (Vaswani et al., 2017) dispense with recurrent networks and convolution.
- Instead they construct both encoders and decoders out of stacks of layers that consist of multi-head attention units which provide input to a feed forward network.
- These layers process input sequences simultaneously, in parallel, without considering sequential order.
- The relative positions of the elements of a sequence are represented as additional information.
- The attention driven design of transformers has allowed them to achieve a significant improvement over LSTMs and CNNs, across a wide range of tasks.

Transformers

- Transformers (Vaswani et al., 2017) dispense with recurrent networks and convolution.
- Instead they construct both encoders and decoders out of stacks of layers that consist of multi-head attention units which provide input to a feed forward network.
- These layers process input sequences simultaneously, in parallel, without considering sequential order.
- The relative positions of the elements of a sequence are represented as additional information.
- The attention driven design of transformers has allowed them to achieve a significant improvement over LSTMs and CNNs, across a wide range of tasks.

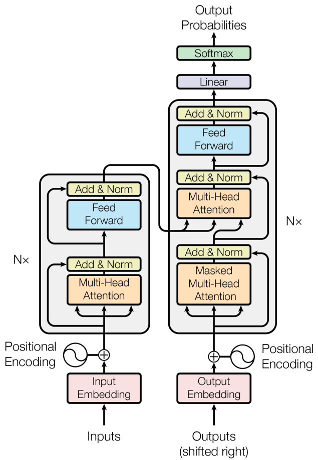
Transformers

- Transformers (Vaswani et al., 2017) dispense with recurrent networks and convolution.
- Instead they construct both encoders and decoders out of stacks of layers that consist of multi-head attention units which provide input to a feed forward network.
- These layers process input sequences simultaneously, in parallel, without considering sequential order.
- The relative positions of the elements of a sequence are represented as additional information.
- The attention driven design of transformers has allowed them to achieve a significant improvement over LSTMs and CNNs, across a wide range of tasks.

Transformers

- Transformers (Vaswani et al., 2017) dispense with recurrent networks and convolution.
- Instead they construct both encoders and decoders out of stacks of layers that consist of multi-head attention units which provide input to a feed forward network.
- These layers process input sequences simultaneously, in parallel, without considering sequential order.
- The relative positions of the elements of a sequence are represented as additional information.
- The attention driven design of transformers has allowed them to achieve a significant improvement over LSTMs and CNNs, across a wide range of tasks.

Architecture of a Transformer



From Ashish Vaswani et al. (2017), "Attention is All you Need", *NIPS 2017*

BERT

- Transformers are pre-trained on large amounts of text for extensive lexical embeddings.
- Many, like OpenAI GPT (Radford et al., 2018) have unidirectional architecture.
- They predict the next element of a sequence given its predecessors, and they do not have access to its successors.
- BERT (Devlin et al., 2019) is a bidirectional transformer trained to predict a masked token from both its left and right contexts (effectively it predicts the word in a blank between two contexts).
- It also uses the same generic parameters from its training for each task to which it is applied, and it is then fine tuned for the particular task.

BERT

- Transformers are pre-trained on large amounts of text for extensive lexical embeddings.
- Many, like OpenAI GPT (Radford et al., 2018) have unidirectional architecture.
- They predict the next element of a sequence given its predecessors, and they do not have access to its successors.
- BERT (Devlin et al., 2019) is a bidirectional transformer trained to predict a masked token from both its left and right contexts (effectively it predicts the word in a blank between two contexts).
- It also uses the same generic parameters from its training for each task to which it is applied, and it is then fine tuned for the particular task.

BERT

- Transformers are pre-trained on large amounts of text for extensive lexical embeddings.
- Many, like OpenAI GPT (Radford et al., 2018) have unidirectional architecture.
- They predict the next element of a sequence given its predecessors, and they do not have access to its successors.
- BERT (Devlin et al., 2019) is a bidirectional transformer trained to predict a masked token from both its left and right contexts (effectively it predicts the word in a blank between two contexts).
- It also uses the same generic parameters from its training for each task to which it is applied, and it is then fine tuned for the particular task.

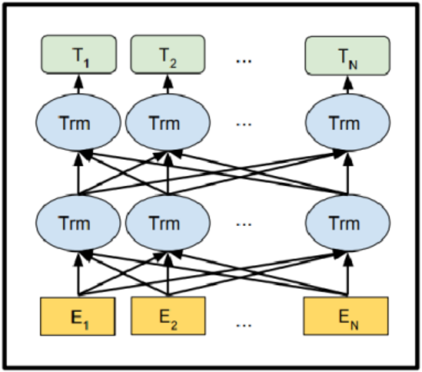
BERT

- Transformers are pre-trained on large amounts of text for extensive lexical embeddings.
- Many, like OpenAI GPT (Radford et al., 2018) have unidirectional architecture.
- They predict the next element of a sequence given its predecessors, and they do not have access to its successors.
- BERT (Devlin et al., 2019) is a bidirectional transformer trained to predict a masked token from both its left and right contexts (effectively it predicts the word in a blank between two contexts).
- It also uses the same generic parameters from its training for each task to which it is applied, and it is then fine tuned for the particular task.

BERT

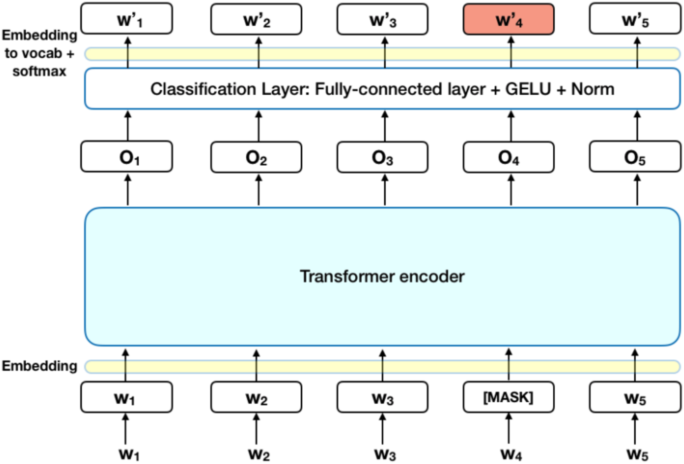
- Transformers are pre-trained on large amounts of text for extensive lexical embeddings.
- Many, like OpenAI GPT (Radford et al., 2018) have unidirectional architecture.
- They predict the next element of a sequence given its predecessors, and they do not have access to its successors.
- BERT (Devlin et al., 2019) is a bidirectional transformer trained to predict a masked token from both its left and right contexts (effectively it predicts the word in a blank between two contexts).
- It also uses the same generic parameters from its training for each task to which it is applied, and it is then fine tuned for the particular task.

BERT's Design



From Jacob Devlin et al. (2019), "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *NAACLT-HLT 2019*

Training BERT



From Rani Horev (2018), "BERT Explained: State of the art language model for NLP", *Towards Data Science*, November 10, 2018

Paraphrase Assessment with DL

- Bizzoni and Lappin (2017) (BL) construct a composite neural network to classify sets of sentences for paraphrase proximity.
- They construct a corpus of 250 sets of five sentences, where each set contains a reference sentence and four paraphrase candidates.
- They rate each of the four candidates on a five point scale for paraphrase proximity to the reference sentence.
- BL train their classifier DNN for both binary and gradient classification of pairs of sentences for paraphrase.
- They train it on 761 pairs of sentences from the corpus, and they test it on 239 pairs.

Paraphrase Assessment with DL

- Bizzoni and Lappin (2017) (BL) construct a composite neural network to classify sets of sentences for paraphrase proximity.
- They construct a corpus of 250 sets of five sentences, where each set contains a reference sentence and four paraphrase candidates.
- They rate each of the four candidates on a five point scale for paraphrase proximity to the reference sentence.
- BL train their classifier DNN for both binary and gradient classification of pairs of sentences for paraphrase.
- They train it on 761 pairs of sentences from the corpus, and they test it on 239 pairs.

Paraphrase Assessment with DL

- Bizzoni and Lappin (2017) (BL) construct a composite neural network to classify sets of sentences for paraphrase proximity.
- They construct a corpus of 250 sets of five sentences, where each set contains a reference sentence and four paraphrase candidates.
- They rate each of the four candidates on a five point scale for paraphrase proximity to the reference sentence.
- BL train their classifier DNN for both binary and gradient classification of pairs of sentences for paraphrase.
- They train it on 761 pairs of sentences from the corpus, and they test it on 239 pairs.

Paraphrase Assessment with DL

- Bizzoni and Lappin (2017) (BL) construct a composite neural network to classify sets of sentences for paraphrase proximity.
- They construct a corpus of 250 sets of five sentences, where each set contains a reference sentence and four paraphrase candidates.
- They rate each of the four candidates on a five point scale for paraphrase proximity to the reference sentence.
- BL train their classifier DNN for both binary and gradient classification of pairs of sentences for paraphrase.
- They train it on 761 pairs of sentences from the corpus, and they test it on 239 pairs.

Paraphrase Assessment with DL

- Bizzoni and Lappin (2017) (BL) construct a composite neural network to classify sets of sentences for paraphrase proximity.
- They construct a corpus of 250 sets of five sentences, where each set contains a reference sentence and four paraphrase candidates.
- They rate each of the four candidates on a five point scale for paraphrase proximity to the reference sentence.
- BL train their classifier DNN for both binary and gradient classification of pairs of sentences for paraphrase.
- They train it on 761 pairs of sentences from the corpus, and they test it on 239 pairs.

The Design of BL's Paraphrase Classifier

The paraphrase classifier consists of three main components:

1. Two encoders, one for each of the sentences in a reference sentence-candidate pair, that consist of a CNN, a max pooling layer, and an LSTM,
2. A merge layer that concatenates the sentence vectors which the encoders produce into a single vector, and
3. Several dense, fully connected layers that apply sigmoid functions to generate a softmax distribution for the paraphrase classification relation between the two input sentences.

The Design of BL's Paraphrase Classifier

The paraphrase classifier consists of three main components:

1. Two encoders, one for each of the sentences in a reference sentence-candidate pair, that consist of a CNN, a max pooling layer, and an LSTM,
2. A merge layer that concatenates the sentence vectors which the encoders produce into a single vector, and
3. Several dense, fully connected layers that apply sigmoid functions to generate a softmax distribution for the paraphrase classification relation between the two input sentences.

The Design of BL's Paraphrase Classifier

The paraphrase classifier consists of three main components:

1. Two encoders, one for each of the sentences in a reference sentence-candidate pair, that consist of a CNN, a max pooling layer, and an LSTM,
2. A merge layer that concatenates the sentence vectors which the encoders produce into a single vector, and
3. Several dense, fully connected layers that apply sigmoid functions to generate a softmax distribution for the paraphrase classification relation between the two input sentences.

The Design of the Encoder

- The DNN uses the pre-trained lexical embeddings of Word2Vec (Mikolov et al., 2013).
- The CNN of the encoder identifies relevant features of an input sentence for the classification task.
- The max pooling layer reduces the dimensions of the vector that the CNN generates.
- The LSTM uses the sequential structure of the sentence vector to highlight features needed for the task, and to further reduce the dimensionality of the input vector.
- The LSTM produces a vector that is passed to two fully connected layers, the first one with a .5 dropout rate (output from half the neurons, randomly selected, of this layer is discarded in training, to avoid overfitting).

The Design of the Encoder

- The DNN uses the pre-trained lexical embeddings of Word2Vec (Mikolov et al., 2013).
- The CNN of the encoder identifies relevant features of an input sentence for the classification task.
- The max pooling layer reduces the dimensions of the vector that the CNN generates.
- The LSTM uses the sequential structure of the sentence vector to highlight features needed for the task, and to further reduce the dimensionality of the input vector.
- The LSTM produces a vector that is passed to two fully connected layers, the first one with a .5 dropout rate (output from half the neurons, randomly selected, of this layer is discarded in training, to avoid overfitting).

The Design of the Encoder

- The DNN uses the pre-trained lexical embeddings of Word2Vec (Mikolov et al., 2013).
- The CNN of the encoder identifies relevant features of an input sentence for the classification task.
- The max pooling layer reduces the dimensions of the vector that the CNN generates.
- The LSTM uses the sequential structure of the sentence vector to highlight features needed for the task, and to further reduce the dimensionality of the input vector.
- The LSTM produces a vector that is passed to two fully connected layers, the first one with a .5 dropout rate (output from half the neurons, randomly selected, of this layer is discarded in training, to avoid overfitting).

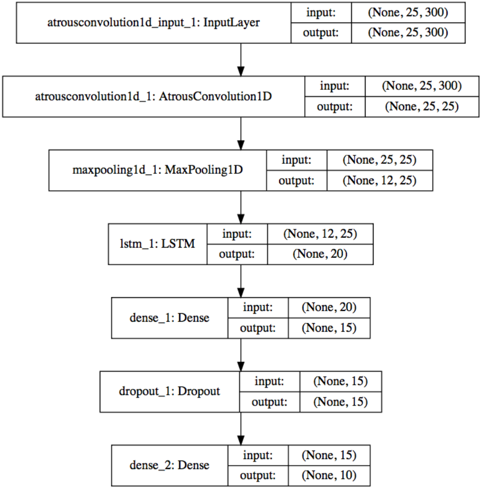
The Design of the Encoder

- The DNN uses the pre-trained lexical embeddings of Word2Vec (Mikolov et al., 2013).
- The CNN of the encoder identifies relevant features of an input sentence for the classification task.
- The max pooling layer reduces the dimensions of the vector that the CNN generates.
- The LSTM uses the sequential structure of the sentence vector to highlight features needed for the task, and to further reduce the dimensionality of the input vector.
- The LSTM produces a vector that is passed to two fully connected layers, the first one with a .5 dropout rate (output from half the neurons, randomly selected, of this layer is discarded in training, to avoid overfitting).

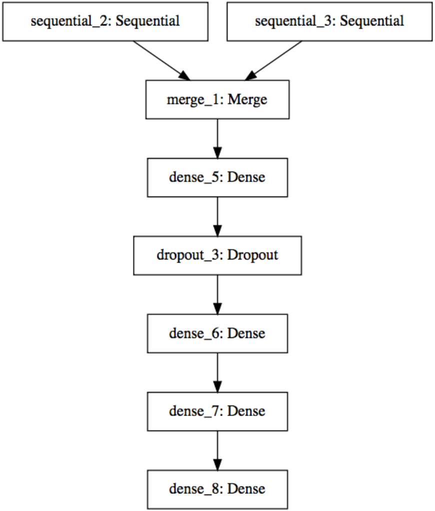
The Design of the Encoder

- The DNN uses the pre-trained lexical embeddings of Word2Vec (Mikolov et al., 2013).
- The CNN of the encoder identifies relevant features of an input sentence for the classification task.
- The max pooling layer reduces the dimensions of the vector that the CNN generates.
- The LSTM uses the sequential structure of the sentence vector to highlight features needed for the task, and to further reduce the dimensionality of the input vector.
- The LSTM produces a vector that is passed to two fully connected layers, the first one with a .5 dropout rate (output from half the neurons, randomly selected, of this layer is discarded in training, to avoid overfitting).

BL Paraphrase Encoder



BL Paraphrase Classifier



Binary and Gradient Paraphrase Classification

- BL assess the accuracy of both types of classification on the basis of their annotation of the test set sentence pairs on a five point scale.
- The binary classifier takes a softmax prediction of a score above a threshold of 2 as an instance of paraphrase.
- The gradient classifier predicts a paraphrase score from the scale, through the softmax probability distribution over this relation.
- They use the Pearson coefficient to evaluate the correlation between the classifier's scores and the ground truth annotations.
- They apply ten fold cross validation to test the robustness of accuracy and correlation.

Binary and Gradient Paraphrase Classification

- BL assess the accuracy of both types of classification on the basis of their annotation of the test set sentence pairs on a five point scale.
- The binary classifier takes a softmax prediction of a score above a threshold of 2 as an instance of paraphrase.
- The gradient classifier predicts a paraphrase score from the scale, through the softmax probability distribution over this relation.
- They use the Pearson coefficient to evaluate the correlation between the classifier's scores and the ground truth annotations.
- They apply ten fold cross validation to test the robustness of accuracy and correlation.

Binary and Gradient Paraphrase Classification

- BL assess the accuracy of both types of classification on the basis of their annotation of the test set sentence pairs on a five point scale.
- The binary classifier takes a softmax prediction of a score above a threshold of 2 as an instance of paraphrase.
- The gradient classifier predicts a paraphrase score from the scale, through the softmax probability distribution over this relation.
- They use the Pearson coefficient to evaluate the correlation between the classifier's scores and the ground truth annotations.
- They apply ten fold cross validation to test the robustness of accuracy and correlation.

Binary and Gradient Paraphrase Classification

- BL assess the accuracy of both types of classification on the basis of their annotation of the test set sentence pairs on a five point scale.
- The binary classifier takes a softmax prediction of a score above a threshold of 2 as an instance of paraphrase.
- The gradient classifier predicts a paraphrase score from the scale, through the softmax probability distribution over this relation.
- They use the Pearson coefficient to evaluate the correlation between the classifier's scores and the ground truth annotations.
- They apply ten fold cross validation to test the robustness of accuracy and correlation.

Binary and Gradient Paraphrase Classification

- BL assess the accuracy of both types of classification on the basis of their annotation of the test set sentence pairs on a five point scale.
- The binary classifier takes a softmax prediction of a score above a threshold of 2 as an instance of paraphrase.
- The gradient classifier predicts a paraphrase score from the scale, through the softmax probability distribution over this relation.
- They use the Pearson coefficient to evaluate the correlation between the classifier's scores and the ground truth annotations.
- They apply ten fold cross validation to test the robustness of accuracy and correlation.

Binary Accuracy and Gradient Correlation

Ten Fold Cross Validation

k	Accuracy
1	70.10
2	67.01
3	79.38
4	73.20
5	67.01
6	72.92
7	66.67
8	75.79
9	64.21
10	73.68
Average	71

k	Pearson
1	0.51
2	0.63
3	0.59
4	0.62
5	0.61
6	0.72
7	0.59
8	0.67
9	0.54
10	0.67
Average	0.61

Conclusions

- DNNs have become increasingly powerful through the use of multi-headed attention heads, and large scale pre-trained embeddings.
- This has facilitated transfer learning, where a DNN trained for one task can be easily adapted to others with the addition of fine-tuning layers.
- Through attention driven architecture and pre-trained embeddings DNNs have come closer to becoming domain general learning procedures that achieve a high level of performance across several domains, with limited task specific training.

Conclusions

- DNNs have become increasingly powerful through the use of multi-headed attention heads, and large scale pre-trained embeddings.
- This has facilitated transfer learning, where a DNN trained for one task can be easily adapted to others with the addition of fine-tuning layers.
- Through attention driven architecture and pre-trained embeddings DNNs have come closer to becoming domain general learning procedures that achieve a high level of performance across several domains, with limited task specific training.

Conclusions

- DNNs have become increasingly powerful through the use of multi-headed attention heads, and large scale pre-trained embeddings.
- This has facilitated transfer learning, where a DNN trained for one task can be easily adapted to others with the addition of fine-tuning layers.
- Through attention driven architecture and pre-trained embeddings DNNs have come closer to becoming domain general learning procedures that achieve a high level of performance across several domains, with limited task specific training.

Conclusions

- Over the past fifteen years DNNs have moved from a niche technique of machine learning to the leading framework for work in AI.
- DL has achieved rapid progress across a wide range of AI tasks, approaching, and in some cases, surpassing human performance on these tasks.
- It has generated significant advances in several areas of NLP in which more traditional, symbolic methods have not yielded robust wide coverage systems after many years of work.
- These results are of cognitive interest to the extent that they show how it is, in principle, possible to effectively acquire certain types of linguistic knowledge through domain general learning devices.

Conclusions

- Over the past fifteen years DNNs have moved from a niche technique of machine learning to the leading framework for work in AI.
- DL has achieved rapid progress across a wide range of AI tasks, approaching, and in some cases, surpassing human performance on these tasks.
- It has generated significant advances in several areas of NLP in which more traditional, symbolic methods have not yielded robust wide coverage systems after many years of work.
- These results are of cognitive interest to the extent that they show how it is, in principle, possible to effectively acquire certain types of linguistic knowledge through domain general learning devices.

Conclusions

- Over the past fifteen years DNNs have moved from a niche technique of machine learning to the leading framework for work in AI.
- DL has achieved rapid progress across a wide range of AI tasks, approaching, and in some cases, surpassing human performance on these tasks.
- It has generated significant advances in several areas of NLP in which more traditional, symbolic methods have not yielded robust wide coverage systems after many years of work.
- These results are of cognitive interest to the extent that they show how it is, in principle, possible to effectively acquire certain types of linguistic knowledge through domain general learning devices.

Conclusions

- Over the past fifteen years DNNs have moved from a niche technique of machine learning to the leading framework for work in AI.
- DL has achieved rapid progress across a wide range of AI tasks, approaching, and in some cases, surpassing human performance on these tasks.
- It has generated significant advances in several areas of NLP in which more traditional, symbolic methods have not yielded robust wide coverage systems after many years of work.
- These results are of cognitive interest to the extent that they show how it is, in principle, possible to effectively acquire certain types of linguistic knowledge through domain general learning devices.