Natural Language Processing for Law and Social Science

9. Sequence Models and Transformers

Outline

Sequence Data

- The real break-through from deep learning for NLP:
  - moving from bag-of-X representations to sequence representations.

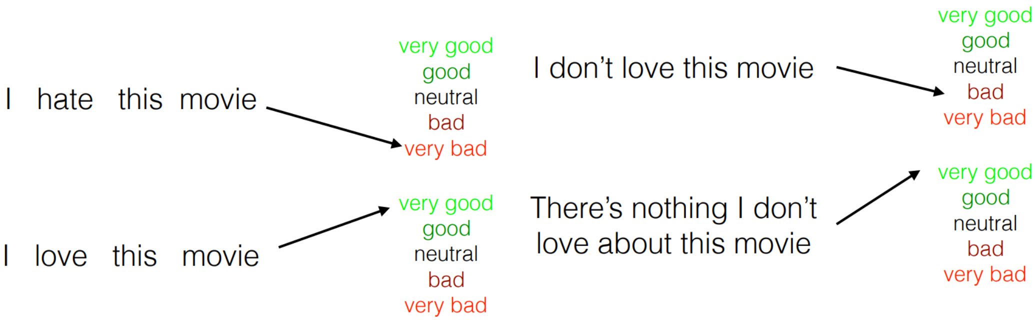
Sequence Data

- The real break-through from deep learning for NLP:
  - moving from bag-of-X representations to sequence representations.
  - Rather than inputting **counts over words** $\boldsymbol{x}$, take as input a **sequence of tokens** $\{w_1, \ldots, w_t, \ldots w_n\}$.

Sequence Data

- The real break-through from deep learning for NLP:
  - moving from bag-of-X representations to sequence representations.
  - Rather than inputting **counts over words** $\boldsymbol{x}$, take as input a **sequence of tokens** $\{w_1, \ldots, w_t, \ldots w_n\}$.
- "Traditional" architectures:
  - Convolutional neural nets (CNNs)
  - Recurrent Neural Nets (RNNs)
- Since 2018, CNNs and RNNs (as currently implemented) usually get worse performance than transformers (attentional neural nets).

The Classic Sentence Classification Problem



Source: Graham Neubig slides.

- bag-of-words models won't capture the importance of "don't love" or "nothing I don't love", even with interactions / hidden layers.
- N-grams have a large feature space (especially with 4-grams) and don't share information across similar words/n-grams.

Convolutional Neural Nets ↔ N-gram Detectors

- A neural net architecture that constructs **filters** that slide across input sequences and extract **local predictive structure**.

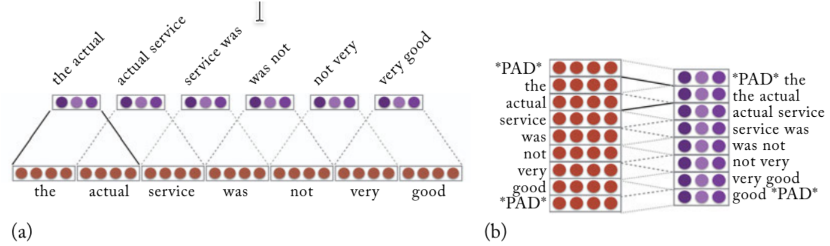


Figure 13.1: The inputs and outputs of a narrow and a wide convolution in the vector-concatenation and the vector-stacking notations. (a) A *narrow* convolution with a window of size $k = 2$ and 3-dimensional output ($\ell = 3$), in the vector-concatenation notation. (b) A *wide* convolution with a window of size $k = 2$, a 3-dimensional output ($\ell = 3$), in the vector-stacking notation.

Johnson and Zhang 2015

- Use CNN for classifying documents by topic and sentiment.
- Interesting part is showing what CNN used for the prediction.

Johnson and Zhang 2015

- Use CNN for classifying documents by topic and sentiment.
- Interesting part is showing what CNN used for the prediction.
- Linear model (SVM) relied on single words and just a few n-grams:
  - poor, useless, returned, not worth, return, worse, disappointed, terrible, worst, horrible
  - great, excellent, perfect, love, easy, amazing, awesome, no problems, perfectly, beat

Johnson and Zhang 2015

- Use CNN for classifying documents by topic and sentiment.
- Interesting part is showing what CNN used for the prediction.
- Linear model (SVM) relied on single words and just a few n-grams:
  - poor, useless, returned, not worth, return, worse, disappointed, terrible, worst, horrible
  - great, excellent, perfect, love, easy, amazing, awesome, no problems, perfectly, beat
- CNN recovers longer, more interesting phrases:

N1	completely useless ., return policy .
N2	it won't even, but doesn't work
N3	product is defective, very disappointing !
N4	is totally unacceptable, is so bad
N5	was very poor, it has failed
P1	works perfectly !, love this product
P2	very pleased !, super easy to, i am pleased
P3	'm so happy, it works perfect, is awesome !
P4	highly recommend it, highly recommended !
P5	am extremely satisfied, is super fast

Table 5: Examples of predictive text regions in the training set.

were unacceptably bad, is abysmally bad, were universally poor, was hugely disappointed, was enormously disappointed, is monumentally frustrating, are endlessly frustrating
best concept ever, best ideas ever, best hub ever, am wholly satisfied, am entirely satisfied, am incredicbly satisfied, 'm overall impressed, am awfully pleased, am exceptionally pleased, 'm entirely happy, are acoustically good, is blindingly fast,

Table 6: Examples of text regions that contribute to prediction. They are from the *test set*, and they did *not* appear in the training set, either entirely or partially as bi-grams.

- but overall, CNNs do not work well in NLP; use embedded hashed n-grams instead (Joulin et al 2016, Goldberg 2017).

RNNs can input and output arbitrary-length sequences

- With previous featurization approaches, documents could be of arbitrary length but the featurizer destroyed any sequence information beyond local word order encoded by n-grams
  - further, they would only output scalars or class probabilities.

RNNs can input and output arbitrary-length sequences

- With previous featurization approaches, documents could be of arbitrary length but the featurizer destroyed any sequence information beyond local word order encoded by n-grams
  - further, they would only output scalars or class probabilities.
- Recurrent Neural Nets (RNNs) work with **sequences of arbitrary length**, both as **inputs** and **outputs**:
  - can *encode* sequences into vectors.
  - can *decode* vectors into sequences.
- therefore especially useful for language tasks such as translation.

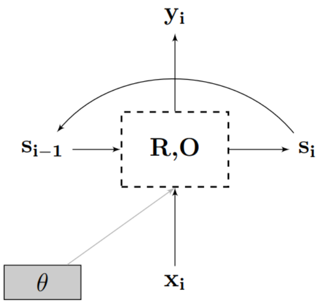
RNN Architecture

- At each step t:
  - a recursion function $R(s_{t-1}, x_t)$ computes the state vector s_t given current word x_t and previous state s_{t-1}.

RNN Architecture

- At each step t:
  - a recursion function $R(s_{t-1}, x_t)$ computes the state vector s_t given current word x_t and previous state s_{t-1}.
  - An output function $O(s_t)$ computes an output vector y_t (to be compared to the outcome variable in the dataset).

$$\hat{\boldsymbol{y}}_t = O(\boldsymbol{s}_t)$$
$$\boldsymbol{s}_t = R(\boldsymbol{s}_{t-1}, \boldsymbol{x}_t)$$



- in a "Simple RNN", $R(\cdot)$ is just a dense layer with ReLU activation, and $O(s_t) = s_t$.

Encoding and Decoding

top left: sequence to sequence; top right: sequence to vector

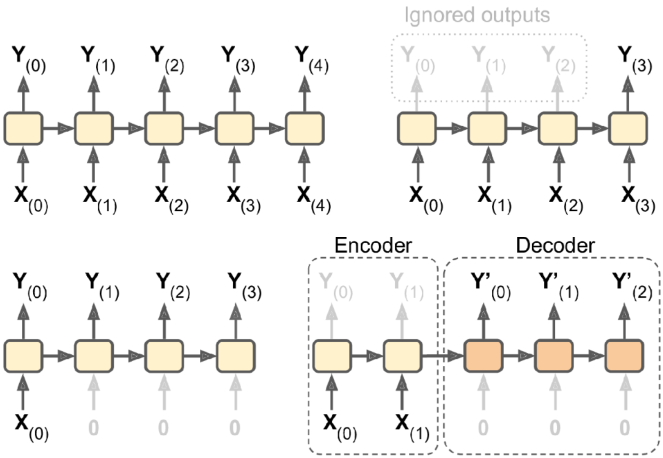
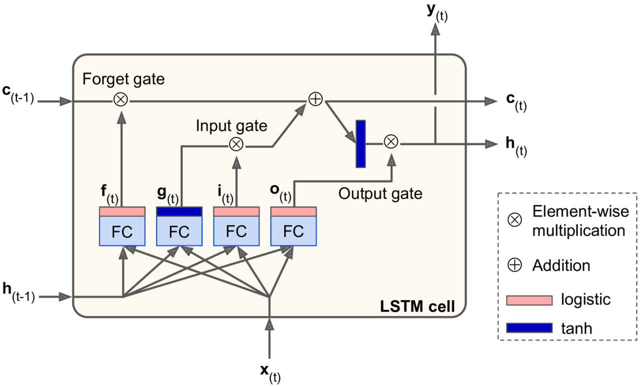


Figure 15-4. Seq-to-seq (top left), seq-to-vector (top right), vector-to-seq (bottom left), and Encoder–Decoder (bottom right) networks

bottom left: vector to sequence; bottom right: encoder-decoder.

Gated Architectures – LSTM (Long Short-Term Memory)



- gating mechanisms prevent vanishing/exploding gradients (see also GRU).
- bidirectional LSTMs (trained backward and forward) get state-of-the-art performance on text classification of short documents (e.g. classifying sentences by sentiment).

RNNs: Practical Use for Sequence-to-Vector Task (see Richard Socher lecture notes)

For example, sentiment analysis:

- tokenize the documents into $\boldsymbol{w}_{1:n}$
- embedding $\boldsymbol{x}_t = \boldsymbol{w}_t \cdot \boldsymbol{\omega}_E$
  - embedding matrix $\boldsymbol{\omega}_E$ can be initialized with pre-trained GloVe embeddings.
- bidirectional LSTM on $\boldsymbol{x}_{\boldsymbol{1:n}}$ (and $\boldsymbol{x}_{\boldsymbol{n:1}}$) to generate document vector $\boldsymbol{s}$
  - that is, document is fed in backwards and forwards to two parallel LSTMs,
  - Use layer normalization (normalize each data point across feature dimensions) rather than batch normalization (normalizes each feature across a sample of data points).
- $\boldsymbol{s}$ input to MLP to predict $\boldsymbol{y}$

Application: RNN's for predicting partisanship

Iyyer, Enns, Boyd-Graber, and Resnik (2014)

Application: RNN's for predicting partisanship

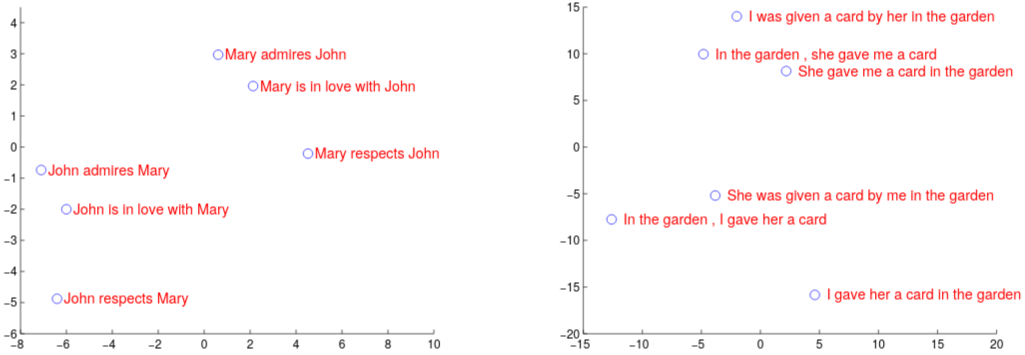
Iyyer, Enns, Boyd-Graber, and Resnik (2014)

n	Most conservative n-grams	Most liberal n-grams
1	Salt, Mexico, housework, speculated, consensus, lawyer, pharmaceuticals, ruthless, deadly, Clinton, redistribution	rich, antipsychotic, malaria, biodiversity, richest, gene, pesticides, desertification, Net, wealthiest, labor, fertilizer, nuclear, HIV
3	prize individual liberty, original liberal idiots, stock market crash, God gives freedom, federal government interference, federal oppression nullification, respect individual liberty, Tea Party patriots, radical Sunni Islamists, Obama stimulus programs	rich and poor,"corporate greed", super rich pay, carrying the rich, corporate interest groups, young women workers, the very rich, for the rich, by the rich, soaking the rich, getting rich often, great and rich, the working poor, corporate income tax, the poor migrants
5	spending on popular government programs, bailouts and unfunded government promises, North America from external threats, government regulations place on businesses, strong Church of Christ convictions, radical Islamism and other threats	the rich are really rich, effective forms of worker participation, the pensions of the poor, tax cuts for the rich, the ecological services of biodiversity, poor children and pregnant women, vacation time for overtime pay
7	government intervention helped make the Depression Great, by God in His image and likeness, producing wealth instead of stunting capital creation, the traditional American values of limited government, trillions of dollars to overseas oil producers, its troubled assets to federal sugar daddies, Obama and his party as racialist fanatics	African Americans and other disproportionately poor groups; the growing gap between rich and poor; the Bush tax cuts for the rich; public outrage at corporate and societal greed; sexually transmitted diseases , most notably AIDS; organize unions or fight for better conditions, the biggest hope for health care reform

Table 2: Highest probability n-grams for conservative and liberal ideologies, as predicted by the **RNN2-(W2V)** model.

RNN's (e.g. Machine Translation) Produce Document Embeddings

- RNN machine translators produce a sentence vector that must be decoded into another language.
- if the vector produces a good translation, it must contain the important information in the sentence.



Sutskever, Vinyals, and Le, "Sequence to sequence learning with neural networks."

Outline

Sentence embeddings

- Document embedding allows us to think of documents as collections of sentences or paragraphs, rather than of words or phrases.
  - would be too high-dimensional to represent each sentence as a vector of word/phrase frequencies.

Sentence embeddings

- Document embedding allows us to think of documents as collections of sentences or paragraphs, rather than of words or phrases.
  - would be too high-dimensional to represent each sentence as a vector of word/phrase frequencies.
- e.g., sentence clustering:
  - run k-means clustering on the dataset of sentences embeddings, then represent documents as counts/frequencies over the sentence clusters.
  - show example sentences to interpret the clusters

Sentence embeddings

- Document embedding allows us to think of documents as collections of sentences or paragraphs, rather than of words or phrases.
  - would be too high-dimensional to represent each sentence as a vector of word/phrase frequencies.
- e.g., sentence clustering:
  - run k-means clustering on the dataset of sentences embeddings, then represent documents as counts/frequencies over the sentence clusters.
  - show example sentences to interpret the clusters
- Sentence mover distance (Clark et al 2019):
  - adapt the idea of word mover distance to sentences.
  - find minimal cost of moving a set of sentence embeddings in document A to co-locate wth a set of sentence embeddings in document B.

Universal Sentence Encoder

```
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

Listing 1: Python example code for using the universal sentence encoder.

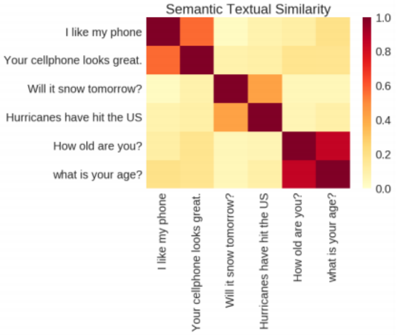


Figure 1: Sentence similarity scores using embeddings from the universal sentence encoder.

- Architecture:
  - Deep Averaging Network with embedded words and bigrams.

Universal Sentence Encoder

```
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

Listing 1: Python example code for using the universal sentence encoder.

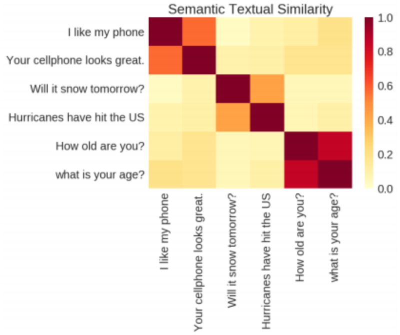


Figure 1: Sentence similarity scores using embeddings from the universal sentence encoder.

- Architecture:
    - Deep Averaging Network with embedded words and bigrams.
- Multiple pre-training objectives:
    - Identifying co-occuring sentences (as in skip thought vectors)
    - Identifying message-response pairs (Henderson et al 2017)
    - Some supervised learning tasks (see Cer et al 2018).

InferSent

- Train a bidirectional LSTM on Stanford Natural Language Inference task:
  - classifying 570K sentence pairs by entailment, contradiction, and neutral.
- The resulting sentence embeddings do better than skip-thought vectors on transfer learning tasks.

Multingual Encoders

- The multilingual sentence encoder (**MUSE**) expands the USE model to sixteen languages, in a single embedding model.
  - Trained on a similar array of tasks in all languages, so that it can be used out-of-the-box.

Multilingual Encoders

- The multilingual sentence encoder (**MUSE**) expands the USE model to sixteen languages, in a single embedding model.
  - Trained on a similar array of tasks in all languages, so that it can be used out-of-the-box.
- Facebook's LASER encoder produces vectors for 90 languages with a single model.
  - bidirectional LSTM architecture
  - trained on multilingual machine translation task

Transformer-Based Sentence Encoders

- The newest version of USE, and the best-performing sentence embeddings overall, use transformer models.
- The standard sentence encoders are in the python package SentenceTransformers.

Outline

Analyzing polarization in social media: Method and application to tweets on 21 mass shootings

Demszky, Garg, Voigt, Zou, Gentzkow, Shapiro, and Jurafsky (2019)

- Dataset:
  - tweets about 21 mass shooting events in USA, 2015-2018.
  - $N = 10{,}000$ (out of 4.4 million tweets from the firehose archive).
  - Party affiliation identified off of whether account follows more Democrats or Republicans
- Text partisanship:
  - measure from Gentzkow, Shapiro, and Taddy (2019) – roughly, text distance between Democrat and Republican twitter accounts.

Sentence Embeddings for Topic Assignment

- Train GloVe embeddings on tweets and create Create Arora et al (2017) embeddings:
- Cluster the embeddings using k-means
- Identify and drop hard-to-classify tweets:
  1. compute ratio of distance to closest topic and distance to second-closest topic.
  2. drop tweets above the 75th percentile.
- Validation using Amazon Mechanical Turk to choose number of clusters:
  - Identify word intruder: five from one cluster, one from another cluster.
  - Identify tweet intruder: three from one cluster, and one from another cluster.

Topic Content

Topic	10 Nearest Stems
news (19%)	break, custodi, #breakingnew, #updat, confirm, fatal, multipl, updat, unconfirm, sever
investigation (9%)	suspect, arrest, alleg, apprehend, custodi, charg, accus, prosecutor, #break, ap
shooter's identity & ideology (11%)	extremist, radic, racist, ideolog, label, rhetor, wing, blm, islamist, christian
victims & location (4%)	bar, thousand, california, calif, among, los, southern, veteran, angel, via
laws & policy (14%)	sensibl, regul, requir, access, abid, #gunreformnow, legisl, argument, allow, #guncontolnow
solidarity (13%)	affect, senseless, ach, heart, heartbroken, sadden, faculti, pray, #prayer, deepest
remembrance (6%)	honor, memori, tuesday, candlelight, flown, vigil, gather, observ, honour, capitol
other (23%)	dude, yeah, eat, huh, gonna, ain, shit, ass, damn, guess

- The embedding method resulted in more coherent topics (better MTurk validation for words and tweets) than a topic model. $k = 8$ got best coherence.
  - Appendix reports samples of tweets for each topic (but does not say how samples were selected).

Between-topic vs within-topic polarization

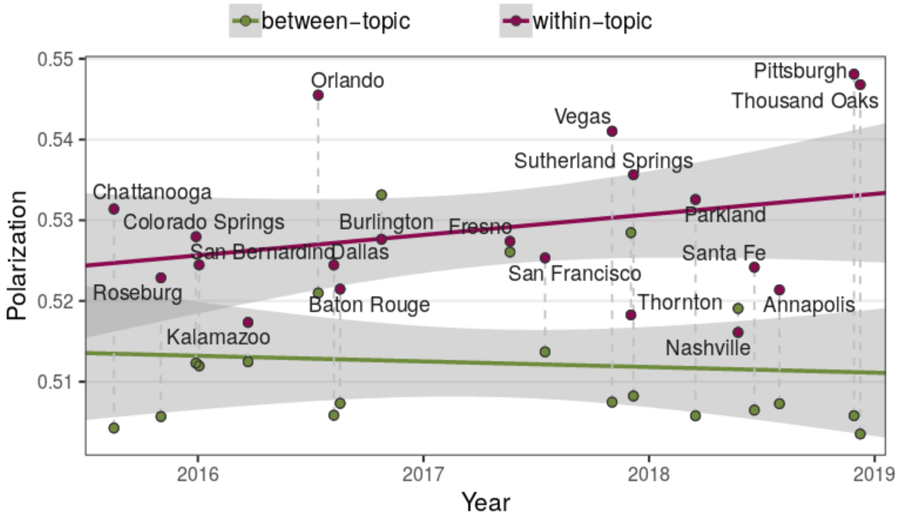
- Within-topic polarization: compute partisan text distance separately by the tweet clusters.

Between-topic vs within-topic polarization

- Within-topic polarization: compute partisan text distance separately by the tweet clusters.
- Between-topic polarization: Compute partisan text distance using cluster counts, rather than token counts.

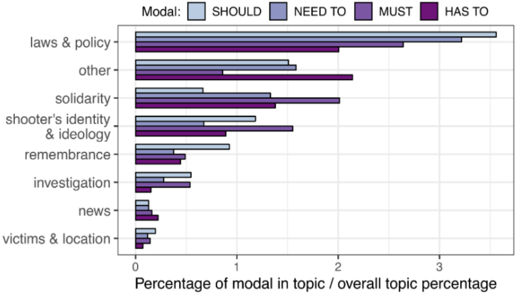
Between-topic vs within-topic polarization

- Within-topic polarization: compute partisan text distance separately by the tweet clusters.
- Between-topic polarization: Compute partisan text distance using cluster counts, rather than token counts.



Modality

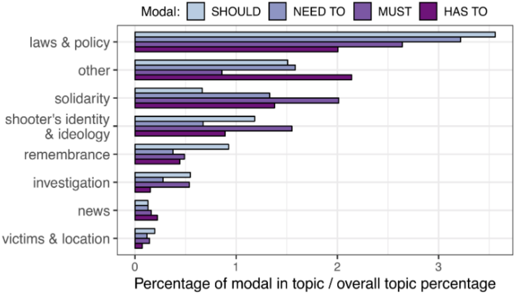
This roller coaster debate **MUST** STOP! Sensible gun ownership is one thing but assault weapons massacre innocent lives. The savagery of gore at #Parkland was beyond belief & **must** be the last.
In times of tragedy **shouldn't** we all come together?! Prayers for those harmed in the #PlannedParenthood shooting.
Communities **need to** step up and address white on white crime like the Las Vegas massacre. White men are out of control.
he BLM protest shooting, planned parenthood, now cali... domestic terrorism will crumble this country, SANE PPL **HAVE TO** FIGHT BACK
Shooting cops is horrible, cannot be condoned. But **must be** understood these incidents are outgrowth of decades of police abuses. #BatonRouge
1. Islamic terrorists are at war with us 2. Gun free zones = kill zones 3. Americans **should be** allowed to defend themselves #Chattanooga
Las Vegas shooting Walmart shooting and now 25 people killed in Texas over 90 people killed Mexico **should** build that wall to keep the US out
CNN reporting 20 dead, 42 injured in Orlando night club shooting. Just awful. The US **must** act to control guns or this carnage will continue.



- Count the four most frequent necessity modals in the data: should, must, have to, need to.
    - in this context, they are used as calls to action.

Modality

This roller coaster debate **MUST** STOP! Sensible gun ownership is one thing but assault weapons massacre innocent lives. The savagery of gore at #Parkland was beyond belief & **must** be the last.
In times of tragedy **shouldn't** we all come together?! Prayers for those harmed in the #PlannedParenthood shooting.
Communities **need to** step up and address white on white crime like the Las Vegas massacre. White men are out of control.
he BLM protest shooting, planned parenthood, now cali... domestic terrorism will crumble this country, SANE PPL **HAVE TO** FIGHT BACK
Shooting cops is horrible, cannot be condoned. But **must be** understood these incidents are outgrowth of decades of police abuses. #BatonRouge
1. Islamic terrorists are at war with us 2. Gun free zones = kill zones 3. Americans **should be** allowed to defend themselves #Chattanooga
Las Vegas shooting Walmart shooting and now 25 people killed in Texas over 90 people killed Mexico **should** build that wall to keep the US out
CNN reporting 20 dead, 42 injured in Orlando night club shooting. Just awful. The US **must** act to control guns or this carnage will continue.



- Count the four most frequent necessity modals in the data: should, must, have to, need to.
  - in this context, they are used as calls to action.
- Democrats use modals more than Republicans; Republicans seem more fatalistic.

Partisanship of Topics, by Race of Shooter

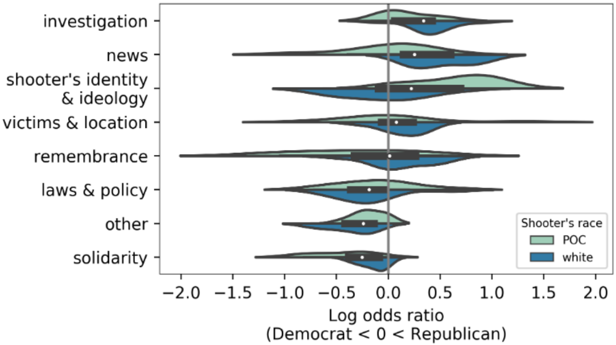


Figure 7: The plot shows the kernel density of the partisan log odds ratios of each topic (one observation per event). The white points show the median and the black rectangles the interquartile range across events.

Partisan Framing Devices: Words

- Partisanship of phrases from supervised model:

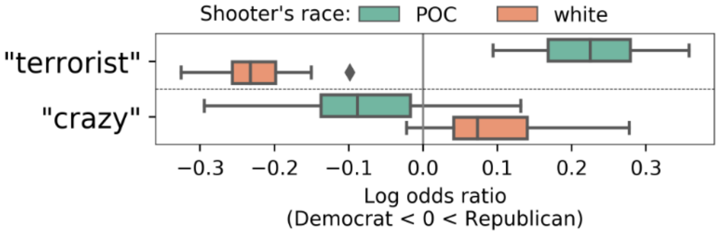


Figure 8: The log odds ratios of "terrorist" and "crazy" across events, grouped by the shooter's race. The boxes show the interquartile range and the diamond an outlier.

- Partisan valence of "terrorist" and "crazy" flip depending on race of shooter (these words have the largest racial difference in the joint vocabulary).

Affect (Emotions)

- Starting point: Emotion lexicon from Mohammad and Turney (2013), available at `saifmohammad.com`.
  - 14,182 words assigned to sentiment (positive/negative) and emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust).

Affect (Emotions)

- Starting point: Emotion lexicon from Mohammad and Turney (2013), available at `saifmohammad.com`.
  - 14,182 words assigned to sentiment (positive/negative) and emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust).
- Domain propagation (Hamilton et al 2018):
  - pick 5-11 representative words per emotion category (Appendix E)
  - for each word in vocabulary, compute average distance to each member of each category. take 30 closest words as lexicon.

Affect (Emotions)

- Starting point: Emotion lexicon from Mohammad and Turney (2013), available at `saifmohammad.com`.
  - 14,182 words assigned to sentiment (positive/negative) and emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust).
- Domain propagation (Hamilton et al 2018):
  - pick 5-11 representative words per emotion category (Appendix E)
  - for each word in vocabulary, compute average distance to each member of each category. take 30 closest words as lexicon.

sadness senseless, loss, tragedi, lost, devast, sad, love, griev, horrif, terribl, pain, violenc, condol, broken, hurt, feel, victim, mourn, horrifi, will, grief, ach, suffer, sick, kill, aw, sicken, evil, massacr, mad

disgust disgust, sick, shame, ignor, wrong, blame, hell, ridicul, idiot, murder, evil, coward, sicken, feel, disgrac, slaughter, action, bad, insan, attack, pathet, outrag, polit, terrorist, mad, damn, lose, shit, lie, asshol

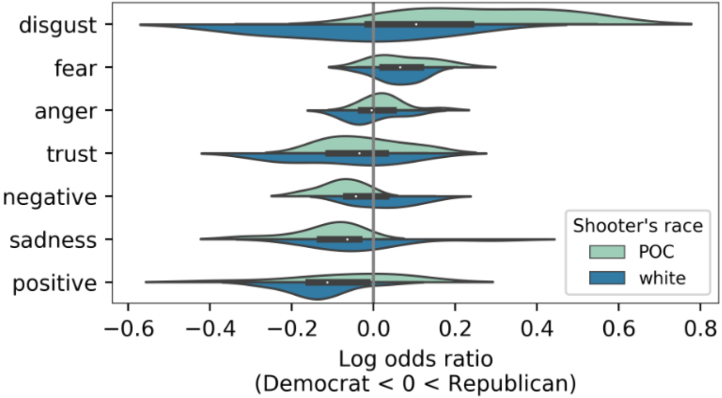
anger gun, will, murder, kill, violenc, wrong, shoot, bad, death, attack, feel, shot, action, arm, idiot, crazi, crimin, terrorist, mad, hell, crime, blame, fight, ridicul, insan, shit, die, threat, terror, hate

fear danger, threat, fear, arm, gun, still, shooter, attack, feel, fight, hide, murder, shot, shoot, bad, kill, chang, serious, violenc, forc, risk, defend, warn, govern, concern, fail, polic, wrong, case, terrorist

trust school, like, good, real, secur, show, nation, don, protect, call, teacher, help, law, great, save, true, wonder, respons, sad, answer, person, feel, safe, thought, continu, love, guard, church, fact, support

Partisanship of Affect Categories

- Compute partisanship scores using affect-category counts:



- Disgust affect flips along partisan lines depending on race of shooter.

We provide an NLP framework to uncover four linguistic dimensions of political polarization in social media: topic choice, framing, affect and illocutionary force. We quantify these aspects with existing lexical methods, and propose clustering of tweet embeddings as a means to identify salient topics for analysis across events; human evaluations show that our approach generates more cohesive topics than traditional LDA-based models. We apply our methods to study 4.4M tweets on 21 mass shootings. We provide evidence that the discussion of these events is highly polarized politically and that this polarization is primarily driven by partisan differences in framing rather than topic choice. We identify framing devices, such as grounding and the contrasting use of the terms "terrorist" and "crazy", that contribute to polarization. Results pertaining to topic choice, affect and illocutionary force suggest that Republicans focus more on the shooter and event-specific facts (news) while Democrats focus more on the victims and call for policy changes. Our work contributes to a deeper understanding of the way group divisions manifest in language and to computational methods for studying them.

1. What is the research question?
2. What is the problem solved?
3. What is being measured?
4. How does the measurement help answer the research question?

Outline

Outline

Attention is All you Need: Transformers

- Since a 2017 paper (Vaswani et al 2017), deep learning for NLP has been transformed by a new class of models: **transformers.**

Attention is All you Need: Transformers

- Since a 2017 paper (Vaswani et al 2017), deep learning for NLP has been transformed by a new class of models: **transformers.**
- Standard approach:
  - represent documents as counts over words/phrases, shares over topics, or the average of word embeddings.

Attention is All you Need: Transformers

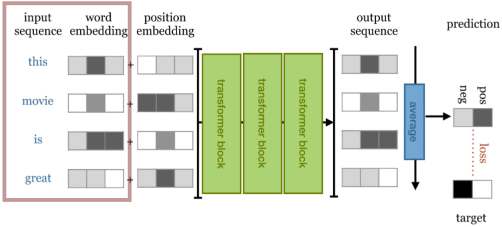
- Since a 2017 paper (Vaswani et al 2017), deep learning for NLP has been transformed by a new class of models: **transformers.**
- Standard approach:
  - represent documents as counts over words/phrases, shares over topics, or the average of word embeddings.
- Recurrent neural nets can process whole documents word-by-word:
  - but they have to sweep through the whole document at each training epoch, so they learn too slowly.

Attention is All you Need: Transformers

- Since a 2017 paper (Vaswani et al 2017), deep learning for NLP has been transformed by a new class of models: **transformers.**
- Standard approach:
  - represent documents as counts over words/phrases, shares over topics, or the average of word embeddings.
- Recurrent neural nets can process whole documents word-by-word:
  - but they have to sweep through the whole document at each training epoch, so they learn too slowly.
- Transformers overcome these limitations:
  - intuitively, they provide a way to efficiently read in an entire document and learn the meaning of all words and all interactions between words.

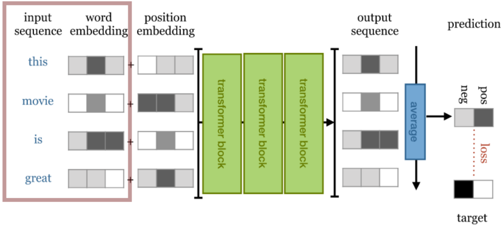
Transformers are powered by ***attention***

- Transformers consist of stacked blocks of parallel **attention heads**
  - analogous to convolutional neural nets, containing stacked blocks of convolutional filters.



Transformers are powered by ***attention***

- Transformers consist of stacked blocks of parallel **attention heads**
  - analogous to convolutional neural nets, containing stacked blocks of convolutional filters.



- **Attention heads** are machine-reading filters, which allow each word to scan over every other word in the document and pick up predictive interactions.

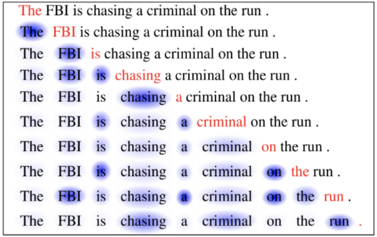


Fig. 6. The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)

OPENAI'S NEW MULTITALENTED AI WRITES, TRANSLATES, AND SLANDERS

A step forward in AI text-generation that also spells trouble

By James Vincent | Feb 14, 2019, 12:00pm EST

Howard, co-founder of Fast.AI agrees. "I've been trying to warn people about this for a while," he says. "We have the technology to totally fill Twitter, email, and the web up with reasonable-sounding, context-appropriate prose, which would drown out all other speech and be impossible to filter."

https://transformer.huggingface.co/doc/distil-gpt2

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in its third version (GPT-3)

GPT and BERT are Pre-Trained Transformer Models

- **GPT = “Generative Pre-Trained Transformer”**:
  - train transformer to predict the next word at the end of a sequence.
  - now in its third version (GPT-3)
  - notably good for text generation (language modeling)

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in its third version (GPT-3)
  - notably good for text generation (language modeling)
- **BERT = "Bidirectional Encoder Representations from Transormers"**:
  - train transformer to predict left-out words in the middle of a sequence.

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in its third version (GPT-3)
  - notably good for text generation (language modeling)
- **BERT = "Bidirectional Encoder Representations from Transormers"**:
  - train transformer to predict left-out words in the middle of a sequence.
  - the resulting document embeddings contain most (perhaps all?) of the relevant information in short language snippets.
    - blew away all the NLP baselines (e.g. semantic role labeling, question-answering, entailment, etc.) when it came out in 2018.

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in its third version (GPT-3)
  - notably good for text generation (language modeling)
- **BERT = "Bidirectional Encoder Representations from Transormers"**:
  - train transformer to predict left-out words in the middle of a sequence.
  - the resulting document embeddings contain most (perhaps all?) of the relevant information in short language snippets.
    - blew away all the NLP baselines (e.g. semantic role labeling, question-answering, entailment, etc.) when it came out in 2018.
- immediately relevant use cases for our purpose:
  - many pre-trained models, e.g. for sentiment classification
  - BERT model can be fine-tuned to quickly get optimal results for many text classification tasks.

Shortcut: Using BERT-Based Pre-Trained Models

Shortcut: Using BERT-Based Pre-Trained Models

```python
from transformers import pipeline
sentiment_analysis = pipeline(“sentiment-analysis”)

pos_text = "I enjoy studying computational algorithms."
neg_text = "I dislike sleeping late everyday."

pos_sent = sentiment_analysis(pos_text)[0]
print(pos_sent['label'],0 pos_sent['score'])

neg_sent = sentiment_analysis(neg_text)[0]
print(neg_sent['label'], neg_sent['score'])
```

- also straightforward to fine-tune BERT for your own classification tasks.
- see notebooks for full details / explanation.

Outline

Self-Attention – the fundamental computation underlying transformers

- Consider a sequence of tokens with fixed length n_L, $\{w_1, ..., w_i, ..., w_{n_L}\}$
- We have word embedding vectors $x_i = E(w_i)$ with dimension n_E, producing a sequence of vectors
$$\{x_1, ..., x_i, ..., x_{n_L}\}$$
- In previous models, the sequence $x_{1:n_L}$ could be flattened to an $n_L n_E$-dimensional vector and piped to the hidden layers for use in the task, e.g. sentiment classification.

Self-Attention – the fundamental computation underlying transformers

- Consider a sequence of tokens with fixed length n_L, $\{w_1, \dots, w_i, \dots, w_{n_L}\}$
- We have word embedding vectors $x_i = E(w_i)$ with dimension n_E, producing a sequence of vectors
$$\{x_1, \dots, x_i, \dots, x_{n_L}\}$$
- In previous models, the sequence $x_{1:n_L}$ could be flattened to an $n_L n_E$-dimensional vector and piped to the hidden layers for use in the task, e.g. sentiment classification.
- A **self-attention layer** transforms $x_{1:n_L}$ into a second sequence $h_{1:n_L}$, where
$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$
  - where $a(\cdot)$ is an attention function such that $a(\cdot) \geq 0$, $\sum a(\cdot) = 1$.
  - $\rightarrow$ each h_i becomes a weighted average of the whole sequence.
- $h_{1:n_L}$ is flattened and piped to the network's hidden layers, rather than $x_{1:n_L}$.

Basic Self-Attention

Setup:

1. Sequence of tokens $\{w_1, ..., w_i, ..., w_{n_L}\}$
2. Sequence of (trainable) embedding vectors $\{x_1, ..., x_i, ..., x_{n_L}\}$
3. Sequence of attention-transformed vectors $\{h_1, ..., h_i, ..., h_{n_L}\}$ with

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Basic Self-Attention

Setup:

1. Sequence of tokens $\{w_1, ..., w_i, ..., w_{n_L}\}$
2. Sequence of (trainable) embedding vectors $\{x_1, ..., x_i, ..., x_{n_L}\}$
3. Sequence of attention-transformed vectors $\{h_1, ..., h_i, ..., h_{n_L}\}$ with

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Basic self-attention specifies

$$a(x_i, x_j) = \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)}$$

- the dot-product $x_i \cdot x_j$, normalized with softmax such that $\sum_j a(\cdot) = 1$.

Basic Self-Attention

Setup:

1. Sequence of tokens $\{w_1, \ldots, w_i, \ldots, w_{n_L}\}$
2. Sequence of (trainable) embedding vectors $\{x_1, \ldots, x_i, \ldots, x_{n_L}\}$
3. Sequence of attention-transformed vectors $\{h_1, \ldots, h_i, \ldots, h_{n_L}\}$ with

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Basic self-attention specifies

$$a(x_i, x_j) = \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)}$$

- the dot-product $x_i \cdot x_j$, normalized with softmax such that $\sum_j a(\cdot) = 1$.
- Putting it together:

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

- The basic self-attention transformation

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

is the foundational ingredient of transformers.

- The basic self-attention transformation

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

  is the foundational ingredient of transformers.

Note the following simplifications:

- **basic self-attention has no learnable parameters**.
  - self-attention works indirectly through the word embeddings (more next slide)
- **basic self-attention ignores word order.**

- The basic self-attention transformation

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

  is the foundational ingredient of transformers.

Note the following simplifications:

- **basic self-attention has no learnable parameters**.
  - self-attention works indirectly through the word embeddings (more next slide)
- **basic self-attention ignores word order.**

The big initial gain from transformers, relative to RNNs, came from basic self-attention.

- The successful models (e.g. BERT, GPT) do add parameters and word order information to $a(\cdot)$ (to be discussed more next week).

Why self-attention works

- Consider a sentence

$$\text{the, cat, walks, on, the, street}$$

  with embeddings

$$\boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{walks}}, \boldsymbol{x}_{\text{on}}, \boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{street}}$$

- Feeding this sentence into the self-attention layer produces

$$\boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{cat}}, \boldsymbol{h}_{\text{walks}}, \boldsymbol{h}_{\text{on}}, \boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{street}}$$

  where $\boldsymbol{h}_i = \sum_{j=1}^{n} \frac{\exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)}{\sum_k \exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_k)} \cdot \boldsymbol{x}_j$.

Why self-attention works

- Consider a sentence

$$\text{the, cat, walks, on, the, street}$$

  with embeddings

$$\boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{walks}}, \boldsymbol{x}_{\text{on}}, \boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{street}}$$

- Feeding this sentence into the self-attention layer produces

$$\boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{cat}}, \boldsymbol{h}_{\text{walks}}, \boldsymbol{h}_{\text{on}}, \boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{street}}$$

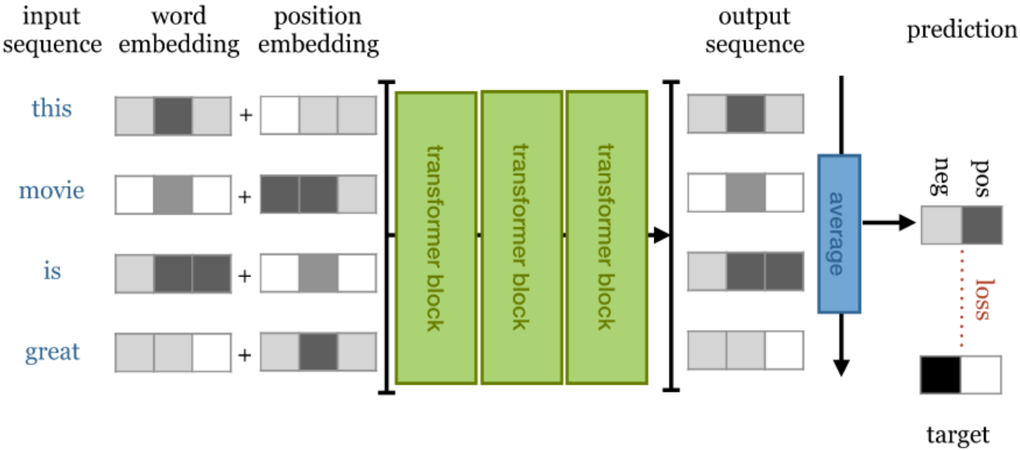
  where $\boldsymbol{h}_i = \sum_{j=1}^{n} \frac{\exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)}{\sum_k \exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_k)} \cdot \boldsymbol{x}_j$.

Embedding layer will learn vectors $\boldsymbol{x}$ that tend to have **attention dot products** that contribute to the task at hand.

- For example, most transformers are pre-trained on a language modeling task (predicting a left-out word or sentence)
- in this task, stopwords like "the" will not be helpful.
  - the learned embedding $\boldsymbol{x}_{\text{the}}$ will tend to have a low or negative dot product with more informative words.

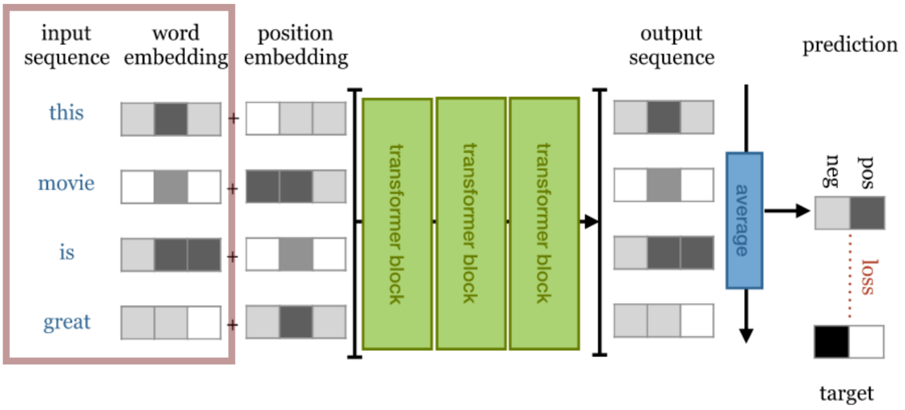
Outline

Transformer for Sentiment Classification



Transformer for Sentiment Classification

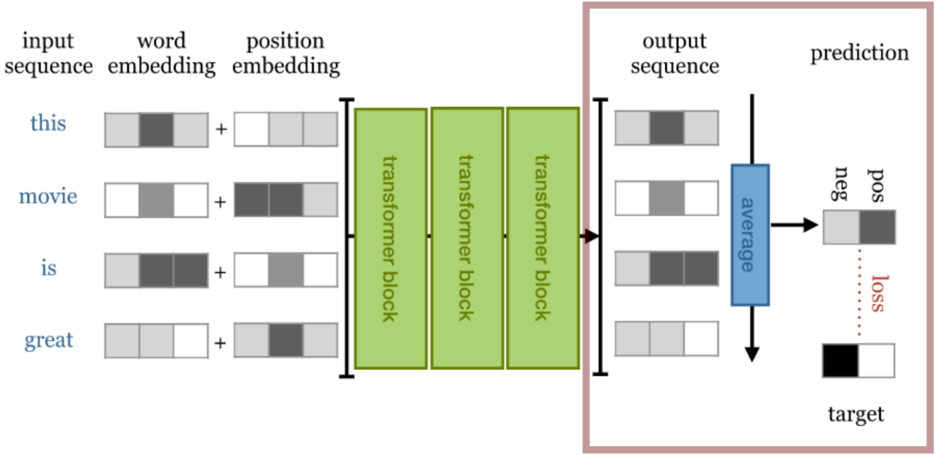
Input sequence → word embedding



- Input sequence of tokens $\{w_1, \ldots, w_i, \ldots, w_{n_L}\}$
- Trainable embedding vectors $[x_1 \ldots, x_i \ldots x_{n_L}]$

Transformer for Sentiment Classification

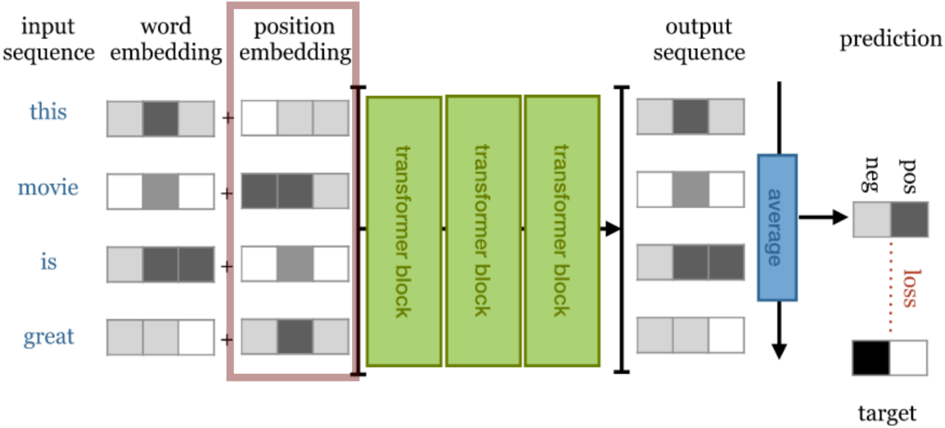
... $\rightarrow$ document embedding $\rightarrow$ sentiment score



- output sequence $\{h_1^y, \ldots, h_i^y, \ldots, h_{n_L}^y\}$
- averaged to produce **document vector** $\vec{d}$
- final output layer with sigmoid activation to produce probabilities $\hat{y}$ across positive and negative output classes.

Transformer for Sentiment Classification

… → position embedding → …



Position Embeddings

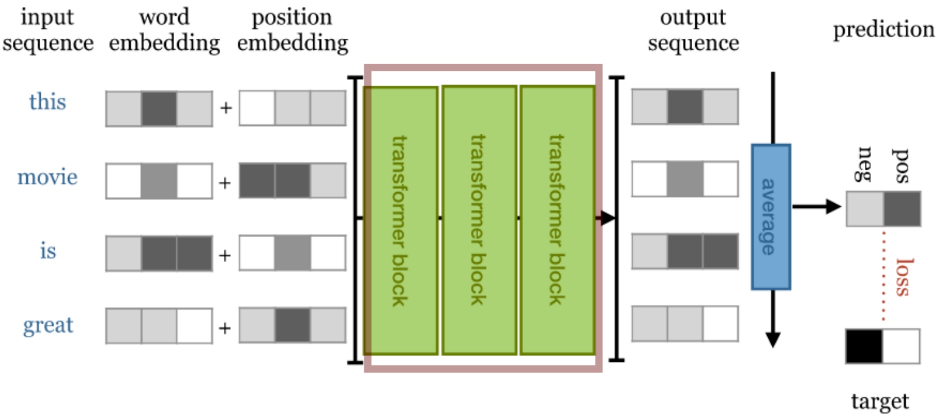
- To add word order information, transformers add a ***position embedding*** along with the ***word embedding*** as input to the attention layer.
- input to transformer block is

$$h^0 = \left[\begin{array}{ccccc} x_1 & \dots & x_i & \dots & x_{n_L} \\ t_1 & \dots & t_i & \dots & t_{n_L} \end{array} \right]$$

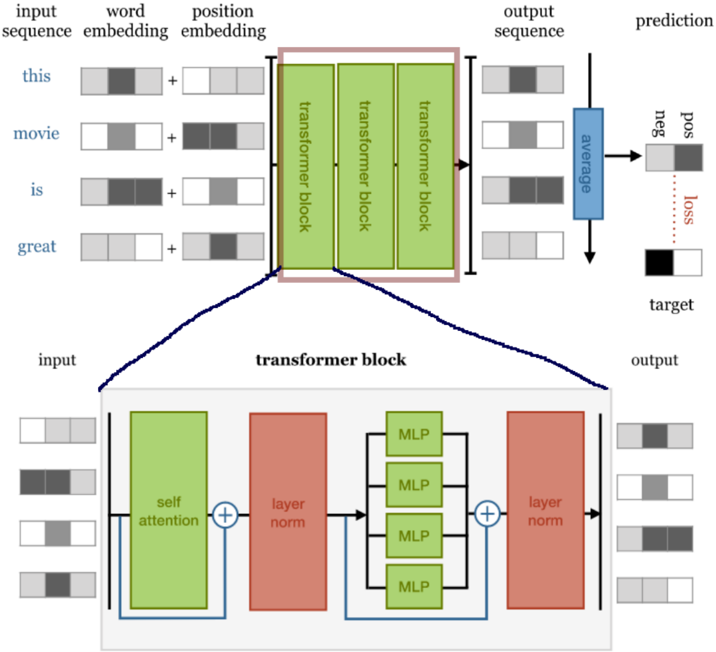
  which includes
  - word embeddings $\{x_1, ..., x_i, ..., x_{n_L}\}$ with dimension n_E
  - stacked with $\{t_1, ..., t_i, ..., t_{n_L}\}$, learnable categorical embeddings with dimension n_t for each index number i itself.
- Note:
  - puts a hard limit on sequence lengths
  - Positional encodings (or any direct information on word order) often not necessary after all (Irie et al 2019; Schlag et al 2021, Sinha et al 2021).

Transformer for Sentiment Classification

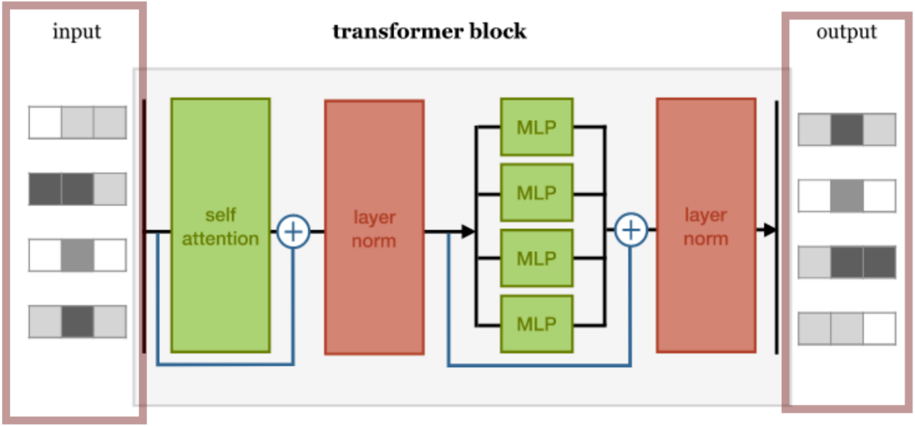
... → transformer blocks → ...



A **transformer** consists of **stacked transformer blocks**

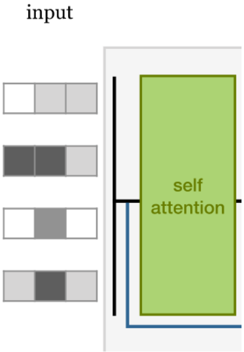


Transformer block (input and output)



- Each transformer block $l \in \{0, \ldots, n_y\}$ takes as input a sequence of vectors $h^l_{1:n_L}$ and outputs a sequence of vectors $h^l_{1:n_L}$, which become the input for the next transformer block.

Transformer Block (Self-Attention Layer)

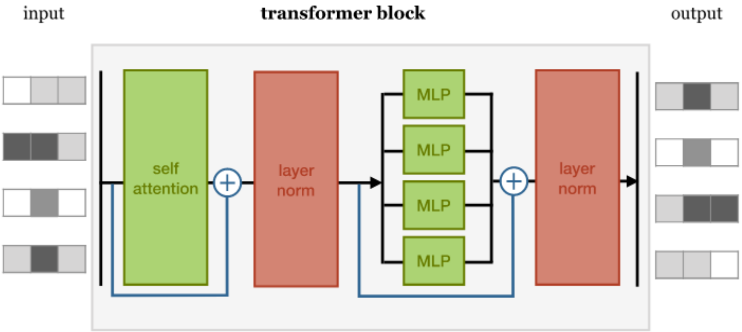


the "self attention" layer:

- input:
  - for the first block, includes the word embeddings and position embeddings h^0
  - for the later blocks, includes the output of the previous block h^l
- output:
  - matrix of self-attention-transformed vectors where item i is

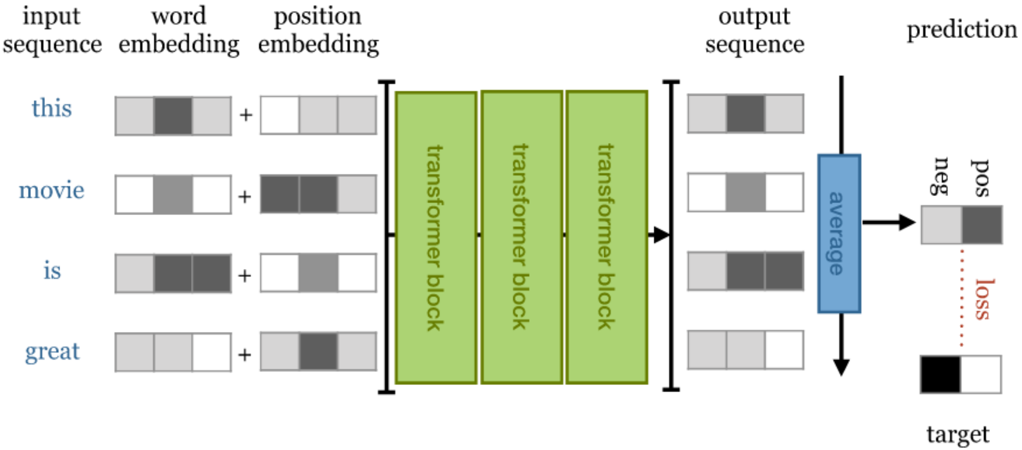
$$\sum_{j=1}^{n_L} a(h_i^l, h_j^l) h_j^l$$

The Transformer Block (Dense Layers)



- self-attention layer's outputs are normalized
  - we will come back to residual connections (blue line with $\oplus$) and "layer normalization" next week.
- piped to a multi-layer perceptron (MLP) with two hidden layers, with ReLU activation after the first layer.
- normalized again then output to h^{l+1}:
  - either to the next transformer block, or to the output layer h^{n_y}.

Transformer for Sentiment Classification



- will get state-of-the-art performance, and much faster to train than a bidirectional LSTM.

Transformers are successful because of scalability

- Attention is not a “better” architecture than recurrence:
  - e.g., a 2-layer transformer is worse than a 2-layer LSTM.

Transformers are successful because of scalability

- Attention is not a “better” architecture than recurrence:
  - e.g., a 2-layer transformer is worse than a 2-layer LSTM.
- Transformers are better thanks to scale.
  - e.g., a 12-layer transformer is better than a 2-layer LSTM

Transformers are successful because of scalability

- Attention is not a “better” architecture than recurrence:
  - e.g., a 2-layer transformer is worse than a 2-layer LSTM.
- Transformers are better thanks to scale.
  - e.g., a 12-layer transformer is better than a 2-layer LSTM
- Transformers are scalable because they are easy to parallelize.
  - With RNNs, training cannot be parallelized because they are connected
    - you need the output at each step to compute the next steps.
  - with transformers, all the blocks can be trained together.

Transformers are successful because of scalability

- Attention is not a “better” architecture than recurrence:
  - e.g., a 2-layer transformer is worse than a 2-layer LSTM.
- Transformers are better thanks to scale.
  - e.g., a 12-layer transformer is better than a 2-layer LSTM
- Transformers are scalable because they are easy to parallelize.
  - With RNNs, training cannot be parallelized because they are connected
    - you need the output at each step to compute the next steps.
  - with transformers, all the blocks can be trained together.
- Thanks to scalability, transformers can become huge and deep and still train efficiently.
  - 12-layer LSTMs do not exist because they are computationally too expensive