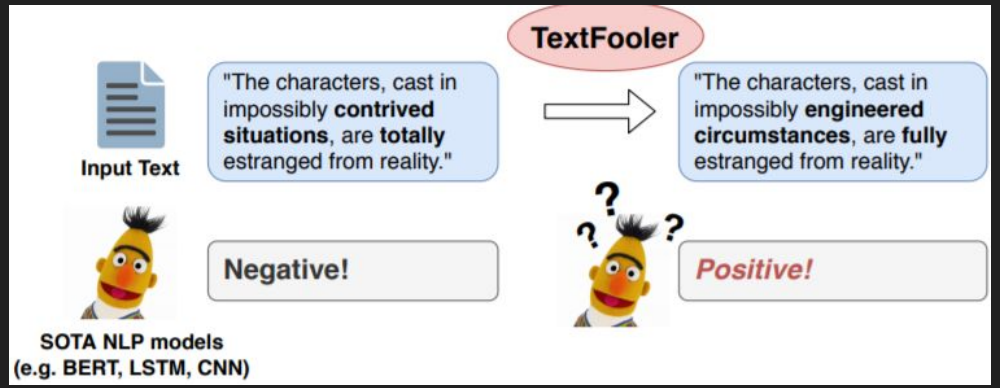
Is BERT Really Robust?

A Strong Baseline for Natural Language Attack on Text Classification and Entailment
Di Jin, Zhijing Jin, Joey Tianyi Zhou, Peter Szolovits

EECS 598 Presentation
03/25/2021
Sahil Farishta

Motivation

- Adversarial text examples are harder to generate
  - Easier for humans to spot differences
  - Previous work has achieved success
    - Typos in various words
    - Insertion or deletion of words
    - All look unnatural to humans
- Three goals
  - Human prediction consistency
  - Semantic similarity
  - Language fluency



Background

- Natural Language Processing involves representing human language in a form machines can understand for machine learning tasks
  - 5 models attacked by this paper
    - BERT (Bidirectional Encoder Representations from Transformers) [2]
      - Considered to be state of the art
      - Looks at words as a whole rather than considering them sequentially
    - Word Based LSTM (Long Short Term Memory) [3]
    - Word Based CNN (Convolutional Neural Net) [4]
    - InferSent (Generalized sentence representations for inference) [5]
    - ESIM (Enhanced LSTM for inference) [6]
- These models generally use an encoder for the read language
  - Use the output as input for a classification or regression task
  - Sample pipeline: English sentence->Encoder->Binary representation->BERT->Classification

Threat Model

- Proposed attack TextFooler
  - Black box attack
    - No access to internal workings (gradients)
  - Has access to model and get predictions and confidence scores
- Given N sentences X = {X1, X2, . . . , XN } and N labels Y = {Y1, Y2, . . . , YN}
  - Model F : X → Y
  - $F(X_{adv}) \neq F(X)$, and $Sim(X_{adv}, X) \geq \varepsilon$ where ε is the minimum similarity score
    - Sim is often defined as a semantic and syntactic similarity function

Algorithm

- Works to find adversarial example for given sentence, label, and target model
  - Uses similarity function and minimum allowed similarity to compute
  - Has access to word embeddings for entire vocabulary
- First finds the importance of certain words and features
  - Allows us to find the most important words to perform the substitution on
- Then we transform words based on the importance score until an acceptable adversarial example is generated
  - Check to make sure words meet criteria
    - Close enough synonyms
    - Correct part of speech
    - Semantic similarity
  - Choose word that can alter the label that’s semantically most similar
    - If no such word exists, choose the word that has the least confidence for the current label

Word Importance Ranking

- Look for importance of words
  - If removing w_i doesn't change label
    - Score is probability change
  - If removing w_i changes label
    - Score is probability change on both original label plus on the new label
- Create finalized set sorted by calculated importance scores
- Black box model involves queries
  - Cannot just look at gradients to determine
- Filter out any stop words from final set
  - NLTK, spaCy

1: Initialization: $X_{\text{adv}} \leftarrow X$
2: **for** each word w_i in X **do**
3: Compute the importance score I_{w_i} via Eq. (2)
4: **end for**
5:
6: Create a set W of all words $w_i \in X$ sorted by the descending order of their importance score I_{w_i}.
7: Filter out the stop words in W.

$$I_{w_i} = \begin{cases} F_Y(X) - F_Y(X_{\setminus w_i}), & \text{if } F(X) = F(X_{\setminus w_i}) = Y \\ (F_Y(X) - F_Y(X_{\setminus w_i})) + (F_{\bar{Y}}(X_{\setminus w_i}) - F_{\bar{Y}}(X)), & \\ \quad \text{if } F(X) = Y, F(X_{\setminus w_i}) = \bar{Y}, \text{ and } Y \neq \bar{Y}. & \end{cases} \quad (2)$$

Equation 2: Importance score calculation

Word Transformer

- Need to look for replacement words
  - Create set of candidates for each word
    - Words are sorted by distance from original
  - Only keep words with same Part-of-Speech (POS)
  - Compute semantic similarity
  - Choose best replacement

for each word w_j in W **do**
  Initiate the set of candidates CANDIDATES by extracting the top N synonyms using $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$ for each word in Vocab.
  CANDIDATES $\leftarrow$ POSFilter(CANDIDATES)
  FINCANDIDATES $\leftarrow$ { }
  for c_k in CANDIDATES **do**
    $X' \leftarrow$ Replace w_j with c_k in X_{adv}
    if $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ **then**
      Add c_k to the set FINCANDIDATES
      $Y_k \leftarrow F(X')$
      $P_k \leftarrow F_{Y_k}(X')$
    end if
  end for
  if there exists c_k whose prediction result $Y_k \neq Y$ **then**
    In FINCANDIDATES, only keep the candidates c_k whose prediction result $Y_k \neq Y$
    $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(X, X'_{w_j \rightarrow c})$
    $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
    return X_{adv}
  else if $P_{Y_k}(X_{\text{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\min} P_k$ **then**
    $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
    $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
  end if
end for
return None

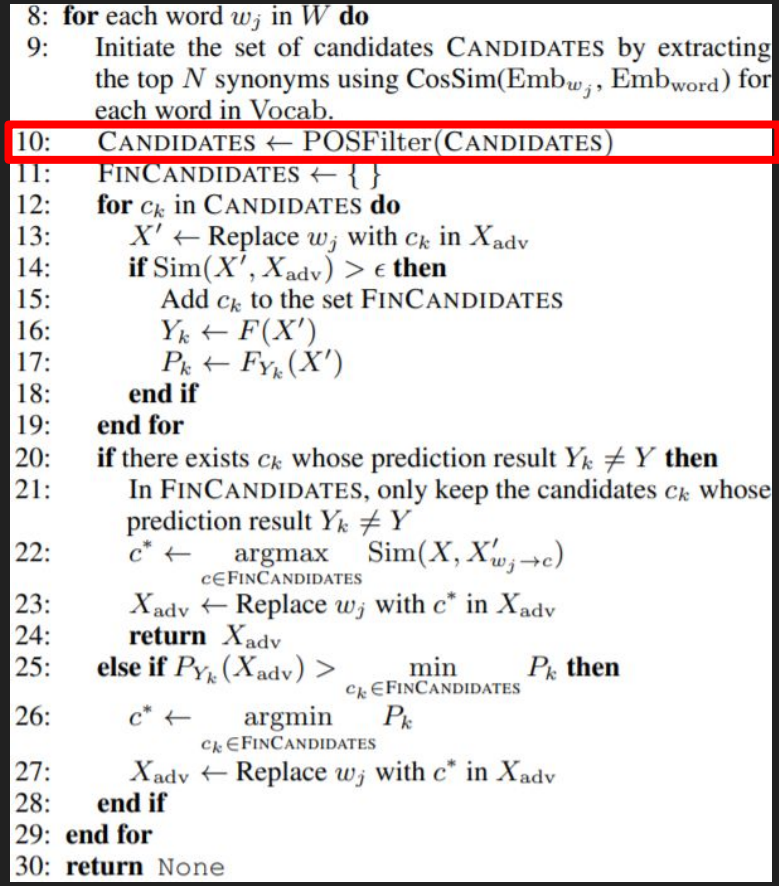
Word Transformer - Sorting

- Words are sorted by distance from original
  - Cosine distance - dot product over norm
  - Use word vectors from SimLex-999
    - Can precompute the matrix
  - Choose top 50 synonyms with similarity > 0.7

8: **for** each word w_j in W **do**
9: Initiate the set of candidates CANDIDATES by extracting the top N synonyms using $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$ for each word in Vocab.
10: CANDIDATES $\leftarrow$ POSFilter(CANDIDATES)
11: FINCANDIDATES $\leftarrow$ { }
12: **for** c_k in CANDIDATES **do**
13: $X' \leftarrow$ Replace w_j with c_k in X_{adv}
14: **if** $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ **then**
15: Add c_k to the set FINCANDIDATES
16: $Y_k \leftarrow F(X')$
17: $P_k \leftarrow F_{Y_k}(X')$
18: **end if**
19: **end for**
20: **if** there exists c_k whose prediction result $Y_k \neq Y$ **then**
21: In FINCANDIDATES, only keep the candidates c_k whose prediction result $Y_k \neq Y$
22: $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(X, X'_{w_j \rightarrow c})$
23: $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
24: **return** X_{adv}
25: **else if** $P_{Y_k}(X_{\text{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\text{min}} P_k$ **then**
26: $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
27: $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
28: **end if**
29: **end for**
30: **return** None

Word Transformer - Filtering

- Only keep words with same Part-of-Speech
  - Preserve grammar and syntax
    - Otherwise it can be spotted by humans
  - Uses spaCy tagger to determine



for each word w_j in W **do**
  Initiate the set of candidates CANDIDATES by extracting the top N synonyms using $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$ for each word in Vocab.
  CANDIDATES $\leftarrow$ POSFilter(CANDIDATES)
  FINCANDIDATES $\leftarrow \{\ \}$
  for c_k in CANDIDATES **do**
    $X' \leftarrow$ Replace w_j with c_k in X_{adv}
    if $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ **then**
      Add c_k to the set FINCANDIDATES
      $Y_k \leftarrow F(X')$
      $P_k \leftarrow F_{Y_k}(X')$
    end if
  end for
  if there exists c_k whose prediction result $Y_k \neq Y$ **then**
    In FINCANDIDATES, only keep the candidates c_k whose prediction result $Y_k \neq Y$
    $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(X, X'_{w_j \rightarrow c})$
    $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
    return X_{adv}
  else if $P_{Y_k}(X_{\text{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\min} P_k$ **then**
    $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
    $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
  end if
end for
return None

Word Transformer - Semantic Analysis

- Compute semantic similarity
  - Universal Sentence Encoder [7]
    - Encode sentences into vectors
    - Compute cosine similarity
    - Keep if similarity score > threshold (0.5)
  - Check label and confidence with word replaced

8: **for** each word w_j in W **do**
9: Initiate the set of candidates CANDIDATES by extracting the top N synonyms using $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$ for each word in Vocab.
10: CANDIDATES $\leftarrow$ POSFilter(CANDIDATES)
11: FINCANDIDATES $\leftarrow \{\}$
12: **for** c_k in CANDIDATES **do**
13: $X' \leftarrow$ Replace w_j with c_k in X_{adv}
14: **if** $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ **then**
15: Add c_k to the set FINCANDIDATES
16: $Y_k \leftarrow F(X')$
17: $P_k \leftarrow F_{Y_k}(X')$
18: **end if**
19: **end for**
20: **if** there exists c_k whose prediction result $Y_k \neq Y$ **then**
21: In FINCANDIDATES, only keep the candidates c_k whose prediction result $Y_k \neq Y$
22: $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(X, X'_{w_j \to c})$
23: $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
24: **return** X_{adv}
25: **else if** $P_{Y_k}(X_{\text{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\text{min}} P_k$ **then**
26: $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
27: $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
28: **end if**
29: **end for**
30: **return** None

Word Transformer - Substitution

- If a replacement changes the label, return substitution with greatest similarity
  - Otherwise make the substitution with the greatest impact on the label
    - Continue on to next word
  - There might not be a replacement for every word in the sentence

8: **for** each word w_j in W **do**
9: Initiate the set of candidates CANDIDATES by extracting the top N synonyms using $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$ for each word in Vocab.
10: CANDIDATES $\leftarrow$ POSFilter(CANDIDATES)
11: FINCANDIDATES $\leftarrow \{\}$
12: **for** c_k in CANDIDATES **do**
13: $X' \leftarrow$ Replace w_j with c_k in X_{adv}
14: **if** $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ **then**
15: Add c_k to the set FINCANDIDATES
16: $Y_k \leftarrow F(X')$
17: $P_k \leftarrow F_{Y_k}(X')$
18: **end if**
19: **end for**
20: **if** there exists c_k whose prediction result $Y_k \neq Y$ **then**
21: In FINCANDIDATES, only keep the candidates c_k whose prediction result $Y_k \neq Y$
22: $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(X, X'_{w_j \rightarrow c})$
23: $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
24: **return** X_{adv}
25: **else if** $P_{Y_k}(X_{\text{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\min} P_k$ **then**
26: $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
27: $X_{\text{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
28: **end if**
29: **end for**
30: **return** None

Experiments

- Run attack against different models across 7 datasets
  - 2 types of datasets
    - Classification = determine label given text
      - WordCNN, WordLSTM, Bert
      - 5 datasets used
    - Entailment = determine logical structure based on text
      - Spot contradictions, entailments, and neutral relationships
      - InferSet, ESIM, Bert
      - 2 datasets used

Datasets

- Classification
  - AG = News corpus - Classify as World, Sports, Business, or Sci/Tech news
  - Fake News = Determine if news is fake or real - From Kaggle dataset
  - MR = Movies reviews - Sentiment analysis on sentences
  - IMDB = Movies reviews - Sentiment analysis on documents
  - Yelp = Yelp reviews - Positive or negative analysis (1 or 2 stars is neg and 4 or 5 stars is pos)
- Textual entailment
  - SNLI (Stanford Natural Language Inference) = Determine relationship between two sentences
    - Example: “The man inspects a uniform” and “The man is sleeping” -> contradiction
  - MultiNLI (Multi Genre Natural Language Inference) = More diverse and complex examples

Automatic Evaluation

- Use metrics to determine effect of attack
  - Original accuracy (before attack)
  - After attack accuracy
  - Percent of words perturbed to create adversarial example
  - Semantic similarity
  - Number of queries made

Human Evaluation

- Evaluate semantic similarity, grammaticality, classification accuracy
  - 100 examples from WordLSTM on MR and 100 from BERT on SNLI
  - Have experts rate grammar of sentences on Likert scale of adversarial and real examples
  - Manually classify both adversarial and real examples and compare consistency
  - Determine similarity of the two examples
  - Found similarity of 0.91 on MR and 0.86 on SNLI
    - 1 is considered identical, 0.5 ambiguous, 0 dissimilar
  - Label agreement score of 0.92 on MR and 0.85 on SNLI
  - Similar grammar scores
    - Adversarial examples have similar grammar correctness as real examples

Source Text	MR (WordLSTM)	SNLI (BERT)
Original	4.22	4.50
Adversarial	4.01	4.27

Grammar scores on Likert Scale

Results - Classification

	WordCNN					WordLSTM					BERT				
	MR	**IMDB**	**Yelp**	**AG**	**Fake**	**MR**	**IMDB**	**Yelp**	**AG**	**Fake**	**MR**	**IMDB**	**Yelp**	**AG**	**Fake**
Original Accuracy	78.0	89.2	93.8	91.5	96.7	80.7	89.8	96.0	91.3	94.0	86.0	90.9	97.0	94.2	97.8
After-Attack Accuracy	2.8	0.0	1.1	1.5	15.9	3.1	0.3	2.1	3.8	16.4	11.5	13.6	6.6	12.5	19.3
% Perturbed Words	14.3	3.5	8.3	15.2	11.0	14.9	5.1	10.6	18.6	10.1	16.7	6.1	13.9	22.0	11.7
Semantic Similarity	0.68	0.89	0.82	0.76	0.82	0.67	0.87	0.79	0.63	0.80	0.65	0.86	0.74	0.57	0.76
Query Number	123	524	487	228	3367	126	666	629	273	3343	166	1134	827	357	4403
Average Text Length	20	215	152	43	885	20	215	152	43	885	20	215	152	43	885

- Accuracy drops immensely after attack
  - No model has higher than 20% accuracy
  - BERT most robust of tested models
  - Fake News dataset hardest to manipulate
    - Easier to make an article look fake than a fake article look real
  - High similarity with usually 10-20% of words perturbed

Results - Entailment

	InferSent		ESIM		BERT	
	SNLI	MultiNLI (m/mm)	SNLI	MultiNLI (m/mm)	SNLI	MultiNLI (m/mm)
Original Accuracy	84.3	70.9/69.6	86.5	77.6/75.8	89.4	85.1/82.1
After-Attack Accuracy	3.5	6.7/6.9	5.1	7.7/7.3	4.0	9.6/8.3
% Perturbed Words	18.0	13.8/14.6	18.1	14.5/14.6	18.5	15.2/14.6
Semantic Similarity	0.50	0.61/0.59	0.47	0.59/0.59	0.45	0.57/0.58
Query Number	57	70/83	58	72/87	60	78/86
Average Text Length	8	11/12	8	11/12	8	11/12

- Similar accuracy scores after attack
  - Lower semantic similarity but fewer queries (smaller texts)

Benchmark Comparison

- Comparison with previous models
  - Li et al. generates misspelled words
  - Alzantot et al. finds perturbations for each word in a sentence
  - Kuleshov et al. performs greedy replacement
- Fewer perturbed words and higher success rate than all models on all tests
  - Previous state of the art algorithms

Dataset	Model	Success Rate	% Perturbed Words
IMDB	(Li et al. 2018)	86.7	6.9
	(Alzantot et al. 2018)	97.0	14.7
	Ours	**99.7**	**5.1**
SNLI	(Alzantot et al. 2018)	70.0	23.0
	Ours	**95.8**	**18.0**
Yelp	(Kuleshov et al. 2018)	74.8	-
	Ours	**97.8**	**10.6**

Reproduced Results

Benchmark	Original Accuracy	Their Post Attack Accuracy	My Post Attack Accuracy
AG	94.2%	12.5%	29%
MR	86.0%	11.5%	10.1%
IMDB	90.9%	13.6%	18.7%

- Rerun on subset of tasks
  - Limited by time (very long training and evaluation times)
  - Used some precomputed values from authors
    - 32GB cosine similarity matrix of fitted word vectors!
- Accuracy is similar to what authors found
  - Exception is AG test which another user on Github reported seeing a similar accuracy after reproducing the results - Limited dataset (1000 examples) provided
  - Datasets used are only a subset of what the authors tested on

Reproduced Results - Thoughts

- Model feels a little heavy too run
  - 32GB worth of computed cosine matrix in addition to the memory needed for the model
- Some generated examples don’t feel convincing on inspection
  - “strange and beautiful film” -> “strange and leggy film”
  - “an interesting slice of history” -> “an interesting lowering of history”
- Model performs well no doubt
  - Even with lower performance in reproduced results, it still cripples the model
  - Especially strong considering it’s a black box attack
  - Room for future improvement
    - Potentially a different similarity score that does not rely on precomputation
    - Would allow for better adaptiveness to new vocabulary

Ablation Study

- Try attack without word importance ranking
  - Instead, randomly perturb examples based on the optimal ratios from the results
  - 45% accuracy increase on average -> important step
- Relaxing semantic similarity constraints
  - Lower accuracy - attacker is less constrained which leads to better adversarial examples
  - Semantically different which allows humans to catch the examples easier
- Some degree of transferability
  - Adversarial examples from SNLI can fool MultiNLI
  - Less so on classification tasks
  - Easier with better model like BERT

Adversarial Training

- Can use generated adversarial examples to retrain model
  - Leads to higher defense against further adversarial attacks
  - Not robust enough
    - Attacks still possible but require perturbing more words

	MR		SNLI	
	Af. Acc.	**Pert.**	**Af. Acc.**	**Pert.**
Original	11.5	16.7	4.0	18.5
+ Adv. Training	**18.7**	**21.0**	**8.3**	**20.1**

Error Analysis

- Three types of errors in generated examples
  - Word sense ambiguity
    - Different synonyms in different contexts
    - Example: “One man shows the ransom money” and “One man testify the ransom money”
  - Grammatical errors
    - Words can have different parts-of-speech depending on context
    - Example: “A man with headphones is biking” and “A man with headphones is motorcycle”
  - Task sensitive content shift
    - Could change the sentence subtly so it means something else entirely
      - Not adversarial anymore because now model is correct to misclassify
    - Example: “A kid in a red hat is running” entails “A kid is running”
      - Switching second sentence to “A girl is running” no longer is an entailment
      - If the model predicts that it’s now a neutral (not entailment or contradiction) the attacker will feel successful
      - But the model is still correct as this is now more accurate

Conclusion

- TextAttack generates adversarial examples in a Black-Box setting that is very effective at paralyzing state-of-the-art NLP models
  - Finds similar words and makes substitutions based on success probability
    - Make take many queries to generate example
  - Text can be a harder domain to work with as humans can more easily spot issues
  - Generated examples are good but have room for improvement
  - Beats previous state-of-the-art attacks

References

1. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. Di Jin, Zhijing Jin, Joey Tianyi Zhou, Peter Szolovits

2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

3. Long-Short Term Memory. Sepp Hochreiter, Jurgen Schmidhuber

4. Convolutional Neural Networks for Sentence Classification. Yoon Kim

5. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, Antoine Bordes

6. Enhanced LSTM for Natural Language Inference. Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, Diana Inkpen

7. Universal Sentence Encoder. Daniel Cer et al.