Skip - Bigrams - Score

1. Bigrams - Two consecutive words in sentence/text
    a. Are all dogs good? - (Are, all), (all, dogs), (dogs, good)
2. Skip - Bigrams - We define a Skip-Bigram to be a pair of terms that either are directly connected in the graph or are both directly connected to a single common (Bskipped[) node in the graph.
3. Score - Let Psb and Qsb be a set of Skip-Bigrams extracted from a passage and a question graph, respectively.
    a.
$$score_P = \frac{|P_{sb} \cap Q_{sb}|}{|P_{sb}|}$$
$$score_Q = \frac{|P_{sb} \cap Q_{sb}|}{|Q_{sb}|}.$$
    b.
$$score = \frac{2 \cdot score_P \cdot score_Q}{score_P + score_Q}.$$

TACS - Textual Alignment Candidate Score

1. Right words and expresses the right content in similar word order as the question
2. TACS handles this well by being robust with respect to differences and providing partial credit for incomplete alignment.
3. The input to the TACS is made of a question string, a passage string, the focus of the question, and the candidate being scored. The output is a similarity measure.
4. Smith-Waterman Algorithm (borrowed from biomed) to figure out the best local alignment score - Kind of DP Max substring ish problem

$$\max \begin{pmatrix} score[i-1][j-1] + \text{sim}(P[i], Q[j]), \\ score[i-1], [j] + \text{sim}(P[i], \phi), \\ score[i][j-1] + \text{sim}(\phi, Q[j]), \\ 0 \end{pmatrix}.$$

Similarity function $\text{sim}(t1, t2)$ is defined as follows:

$$\text{sim}(t1, t2) = \begin{cases} \text{idf}(t_1), & \text{if } t_1 = t_2 \\ -\text{idf}(t_1), & \text{if } t_2 = \phi \\ -\text{idf}(t_2), & \text{if } t_1 = \phi \\ -\text{idf}(t_1), & \text{if otherwise,} \end{cases}$$

5. $\text{sim}(FOCUS, CANDIDATE) = \log(N).$

Jaccard's Similarity

IoU for sets of words (lemmatized)

WMD - Word Mover Distance https://www.youtube.com/watch?v=SqQ7vWrqccA

Noun Phrase A string of words that can be replaced by a single pronoun without rendering the sentence grammatically unacceptable is a noun phrase

Detect noun phrases/phrases of interest from unidirectional links to consecutive words in

dependency graph