Natural Language Processing

Week 1: Modelling Compression with Discourse Constraints

Sentence compression assumes that sentence reduction can be realised in isolation without making use of discourse-level information.

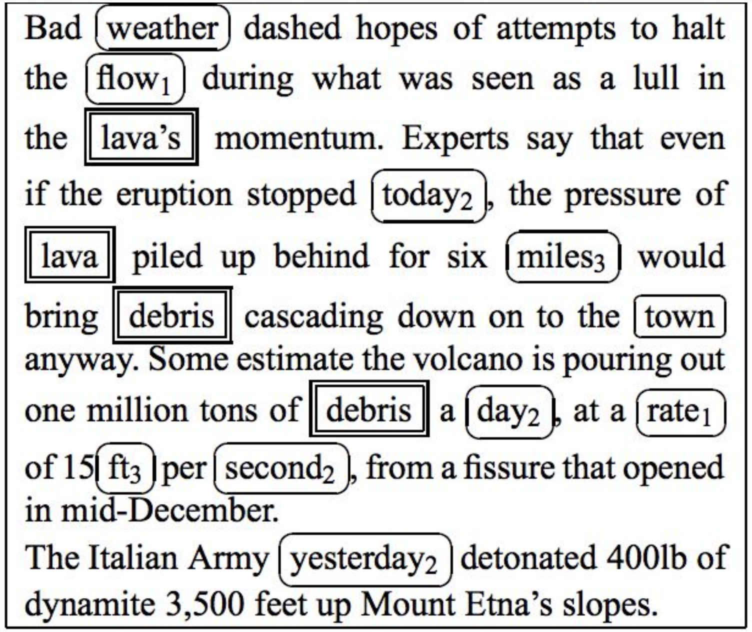
This is clearly not true — professional abstracters often rely on contextual cues while creating summaries.

Then how would a summary using sentence compression be coherent?

1. Centering theory: Assumption that a single entity is salient or *centered.*
    a. Each utterance U_i in a discourse segment has a list of forward-looking centers, $C_f(U_i)$ and a unique backward-looking center, $C_b(U_i)$
    b. Algorithm: Extract entities from U_i => Create $C_f(U_i)$ by ranking the entities in U_i according to their grammatical role (subjects > objects > others) => Find the highest ranked entity in $C_f(U_{i-1})$ which occurs in $C_f(U_i)$, set the entity to be $C_b(U_i)$.

2. Lexical chains: Lexical cohesion refers to the degree of semantic relatedness observed among lexical items in a document.
    a. A document with many different lexical chains will probably contain several topics. And main topics will tend to be represented by dense and long chains.
    b. Compute the lexical chains for the document => Score(Chain) = Sentences(Chain) => Discard chains if Score(Chain) < Avg(Score) => Mark terms from the remaining chains as being the focus of the document

Week 1: Modelling Compression with Discourse Constraints

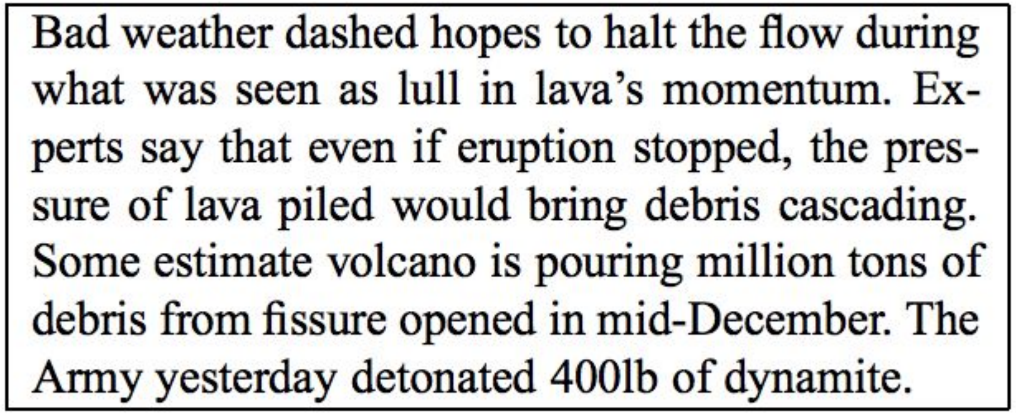


Bad weather dashed hopes of attempts to halt the flow$_1$ during what was seen as a lull in the lava's momentum. Experts say that even if the eruption stopped today$_2$, the pressure of lava piled up behind for six miles$_3$ would bring debris cascading down on to the town anyway. Some estimate the volcano is pouring out one million tons of debris a day$_2$, at a rate$_1$ of 15 ft$_3$ per second$_2$, from a fissure that opened in mid-December.
The Italian Army yesterday$_2$ detonated 400lb of dynamite 3,500 feet up Mount Etna's slopes.

Figure 1: Excerpt of document from our test set with discourse annotations. Centers are in double boxes; terms occurring in lexical chains are in oval boxes. Words with the same subscript are members of the same chain (e.g., *today*, *day*, *second*, *yesterday*)

Compression model: Given a long sentence, a compression is formed by retaining the words that maximise a scoring function.

Optimal subtree problem: Binary decision whether to include each word in the sentence compression using a trigram language model => assign significance scores to each possible compression => evaluate the compression on the basis of constraints (centering and lexical chains to maintain flow)



Bad weather dashed hopes to halt the flow during what was seen as lull in lava's momentum. Experts say that even if eruption stopped, the pressure of lava piled would bring debris cascading. Some estimate volcano is pouring million tons of debris from fissure opened in mid-December. The Army yesterday detonated 400lb of dynamite.

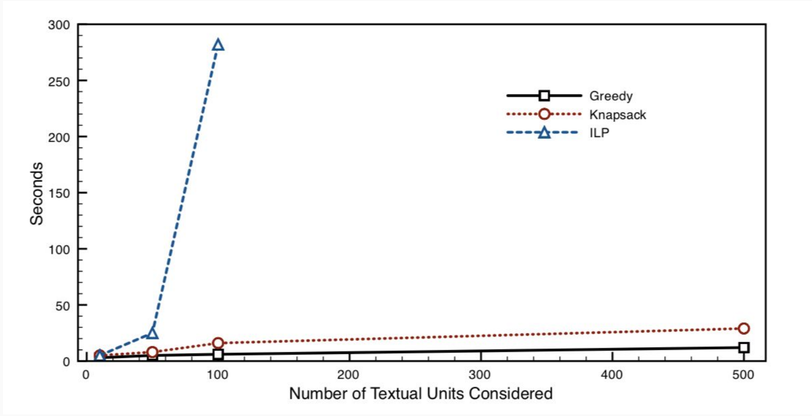
Week 2: A Study of Global Inference Algorithms in Multi-Document Summarization

Summaries from clusters of documents: Relevance, Redundancy, Length - **global inference problem**

1. Solve relevance and redundancy separately - textual units are first clustered and then representative units are chosen by each cluster to be included in the final summary.
2. Optimize all three criteria in tandem - Maximum Marginal Relevance (MMR) that scores each summary sentence and then greedily includes each sentence to maximise the summary score

Three algorithms: (Input = n documents and K is the length of the summary)

1. MMR - Greedy algorithm: Begin by including highly relevant textual units, and then to iteratively add new units that maximize the objective. O(n log n + Kn)
   a. Error propagation: If you include the wrong units early on.
2. Dynamic programming: Fill out a grid by the knapsack solution so as to reach K. Of course, this doesn't account for redundancies so you have to greedily look at combinations of units to arrive at an approximation. O(n log n + Kn)
   a. Accuracy is lost to approximations
3. Integer Linear Programming (ILP): (discussed in first paper) - very computationally intense and scales super-linearly



DP algorithm provides optimal accuracy and scaling.

Probabilistic Language Modelling

Language Model: To compute the probability of a sentence or a sequence of words

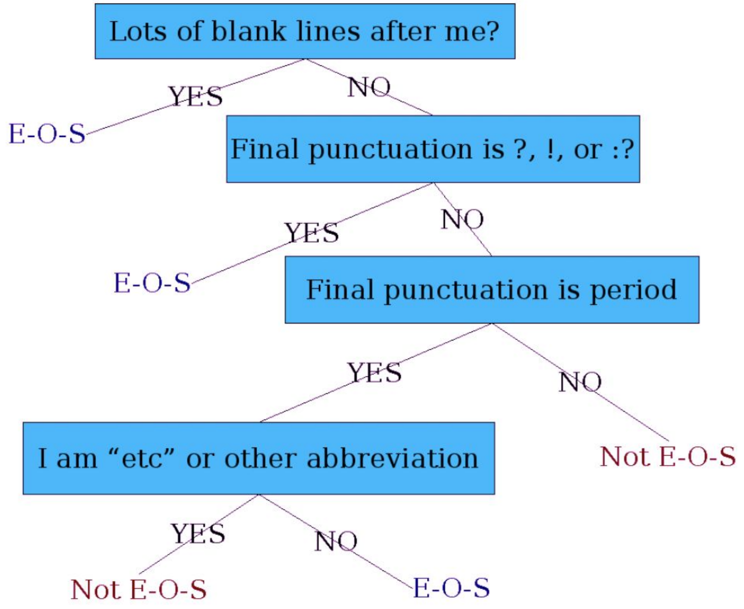
$$
\begin{aligned}
&P(\text{"its water is so transparent"}) = \\
&\quad P(\text{its}) \times P(\text{water} \mid \text{its}) \times P(\text{is} \mid \text{its water}) \\
&\qquad \times P(\text{so} \mid \text{its water is}) \times P(\text{transparent} \mid \text{its water is so})
\end{aligned}
$$

We use the markov assumption to approximate each component in the product in the chain rule of probabilities.

The simplest case is the unigram model and the next case is the bigram model where you condition on the previous word etc. and this is how you get to N-grams. But what about long-distance dependencies?

N-grams only work well for word prediction if the test corpus looks like the training corpus.

Determining if a word is end-of-sentence: a Decision Tree


Lots of blank lines after me?
YES
NO
E-O-S
Final punctuation is ?, !, or :?
YES
NO
E-O-S
Final punctuation is period
YES
NO
I am "etc" or other abbreviation
Not E-O-S
YES
NO
Not E-O-S
E-O-S

Latent Dirichlet Allocation (LDA) - Topic modelling is a problem in information retrieval (2003)

Three-level hierarchical generative Bayesian model - Each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.

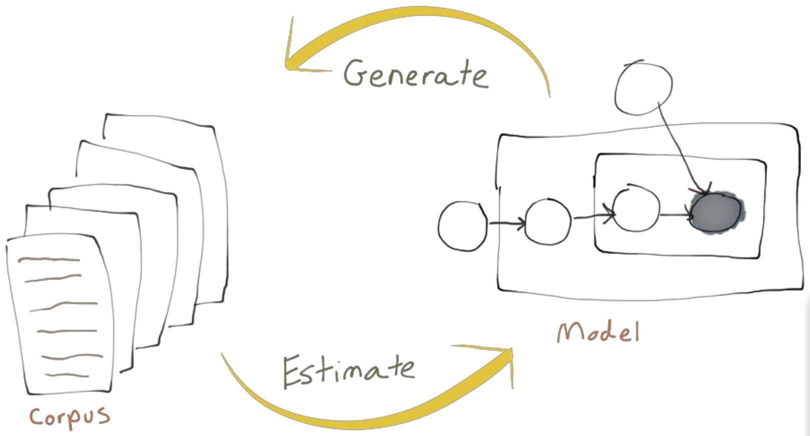
Assumption: Sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently.

Algorithm: (Input - n documents and k topics)

```
1. Go through each document and assign each word to a given topic randomly (topics decided according to dirichlet dist.)
     - This random assignment gives us both topic representations and word distributions of all the topics but we want to
       improve on this randomness
2. For each document d:
   a. For each word w in d:
      i. For each topic t:
         1. Compute p(topic t | document d) which is the proportion of words in document d that are currently
            assigned to topic t
         2. p(word w | topic t)  which is the proportion of assignments to topic t over all documents that come from
            this word w
         3. Reassign w to topic t where t is the highest value of p(topic t | document d) * p(word w | topic t)
              - It's a generative model so we are assuming that topic t generated the word w
              - We also assume that all topic assignments except the current one are correct
   Repeat this until you reach a sufficiently "stable" state aka topic assignments don't change
```

Cool explanation here

Latent Dirichlet Allocation (LDA) - Topic modelling is a problem in information retrieval



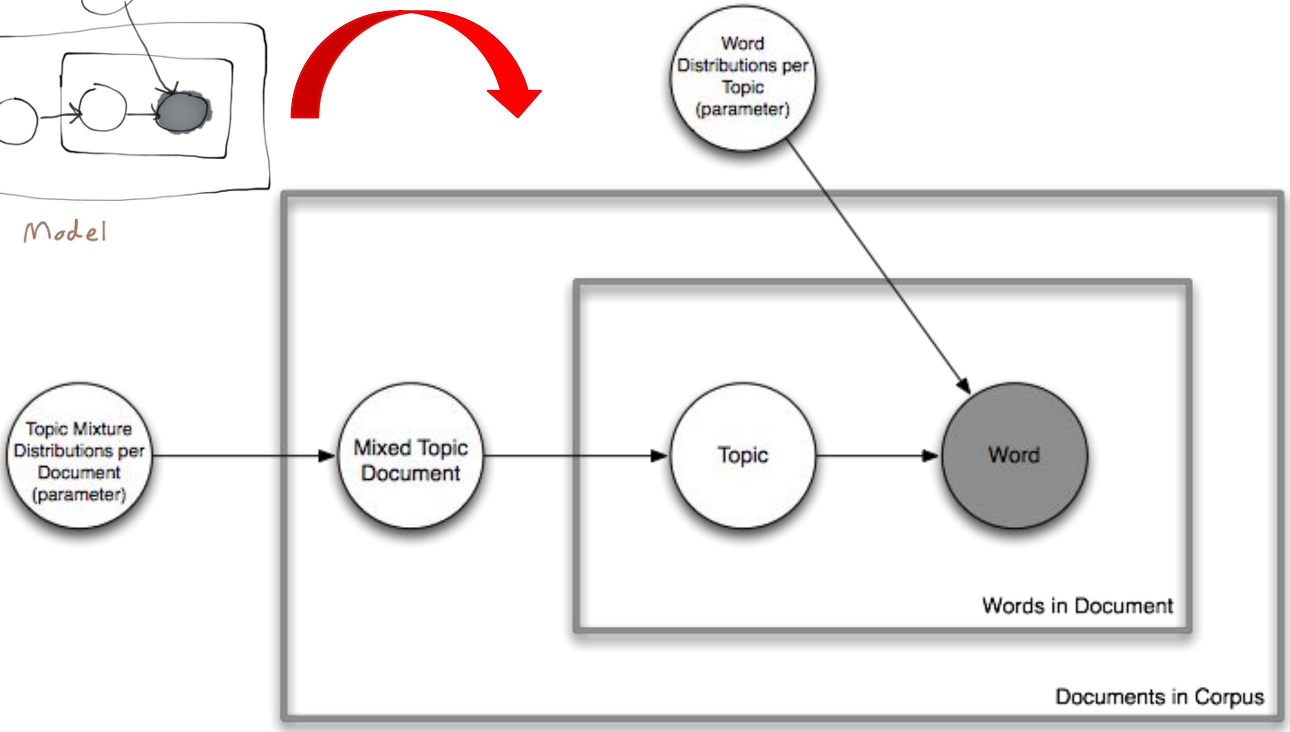
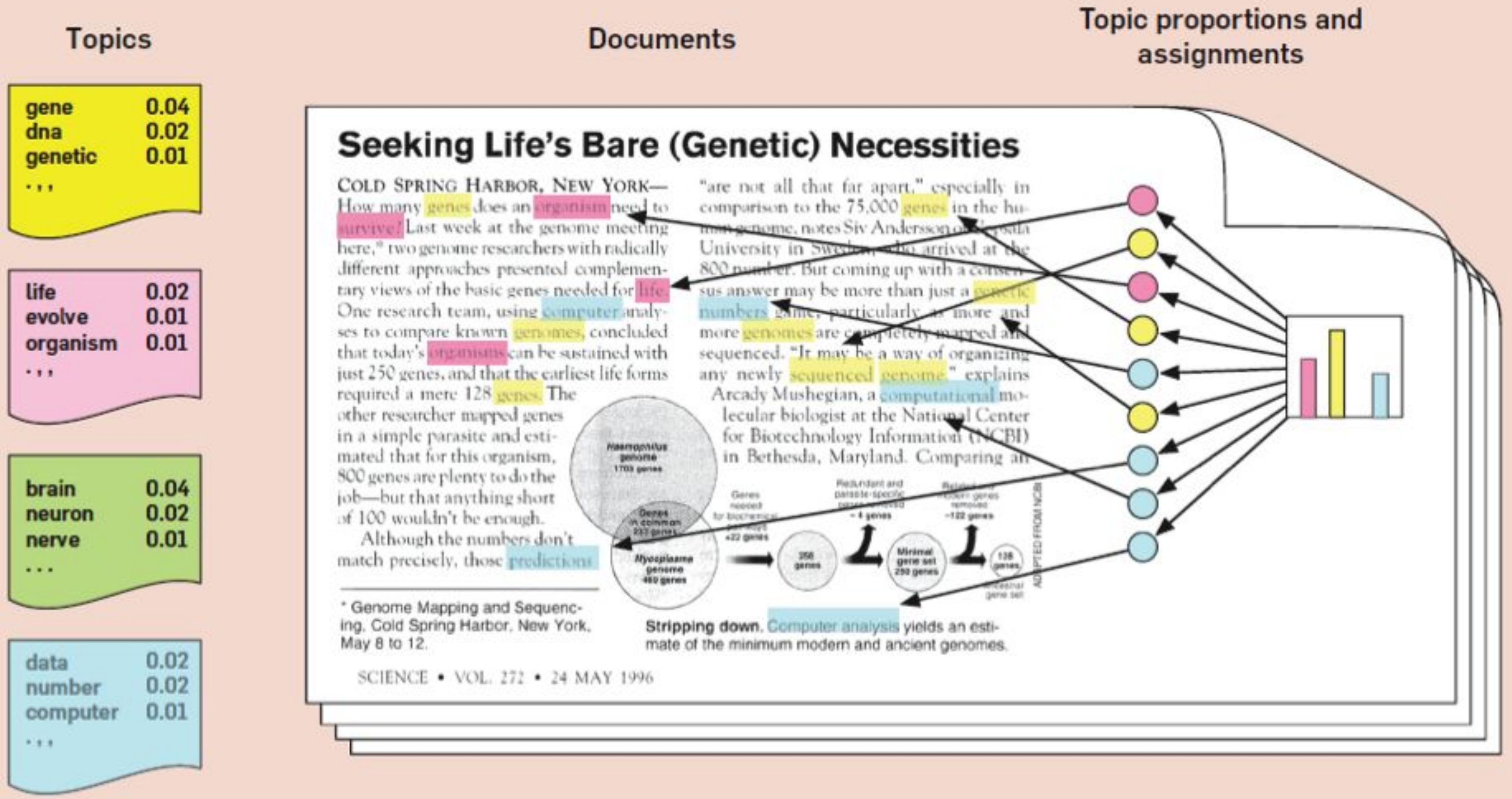

Word Distributions per Topic (parameter)
Topic Mixture Distributions per Document (parameter)
Mixed Topic Document
Topic
Word
Words in Document
Documents in Corpus

Figure 1. The intuitions behind latent Dirichlet allocation. We assume that some number of "topics," which are distributions over words, exist for the whole collection (far left). Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); then, for each word, choose a topic assignment (the colored coins) and choose the word from the corresponding topic. The topics and topic assignments in this figure are illustrative—they are not fit from real data. See Figure 2 for topics fit from data.



Topics
Documents
Topic proportions and assignments
gene 0.04
dna 0.02
genetic 0.01
...
life 0.02
evolve 0.01
organism 0.01
...
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
...
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

A Neural Probabilistic Language Model

Traditional approach: Use n-grams to concatenate short overlapping sequences seen in the training set

Curse of dimensionality: A word sequence on which the model will be tested is likely to be different from all the word sequences seen during training.

Learning a distributed representation for words such that the model learns 1) a distributed representation for each word 2) the probability function for word sequences

1. Associate with each word in the vocabulary a distributed word feature vector.
2. Express the joint probability function of word sequences in terms of the feature vectors of these words in the sequence.
3. Learn simultaneously the word feature vectors and the parameters of that probability function.

Tune 2 to maximize the log-likelihood of the training data.. **why**?

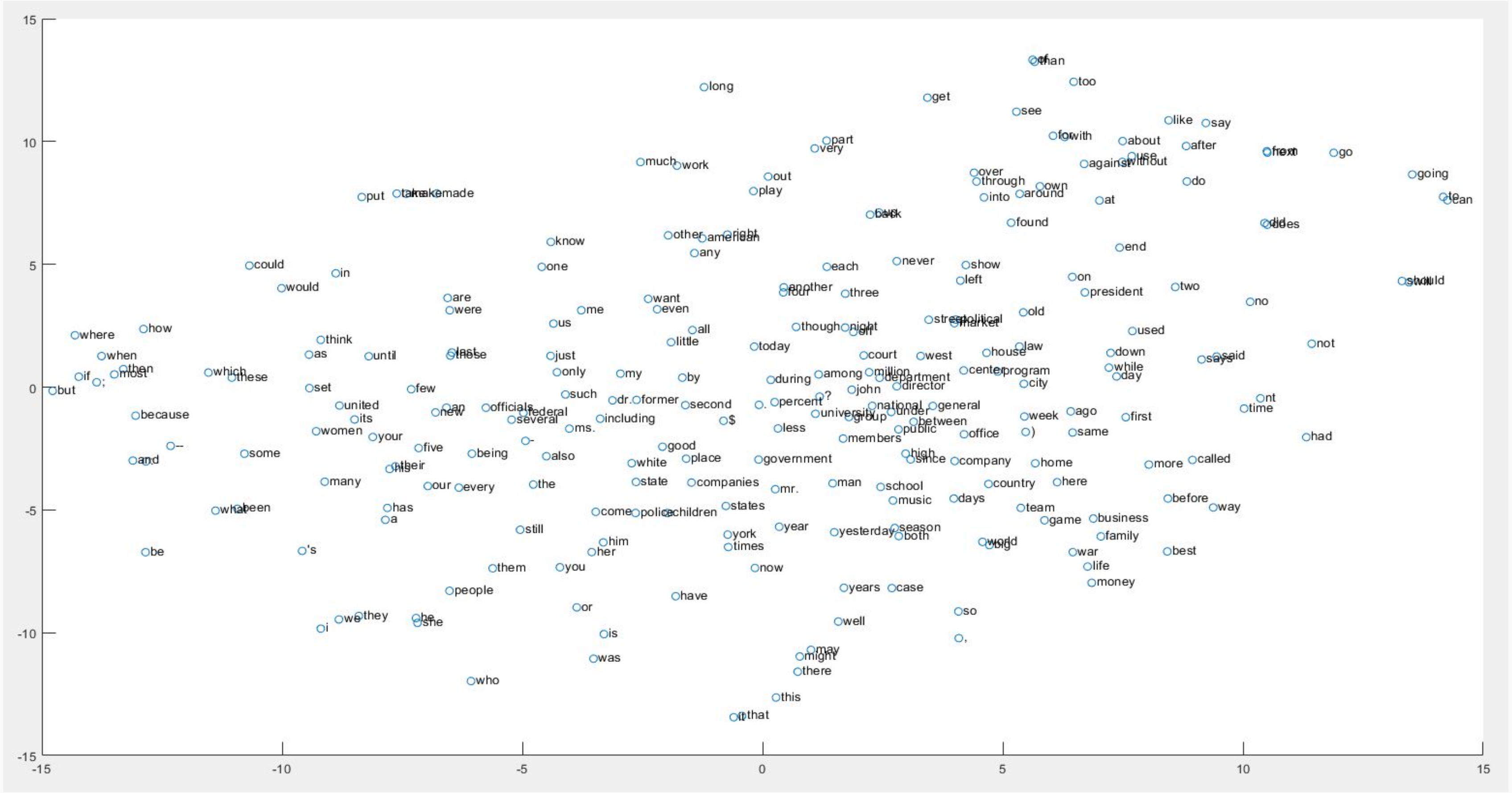
Neural network architecture..page 5

Use a neural network to estimate the conditional probability of the next word given the the distributed representations of the context words. Similar words (distributionally) will have similar feature vectors — small change in feature vector will result in small change in probability estimate (since the NN is a smooth function)

A Neural Probabilistic Language Model

- Train the NN using stochastic gradient ascent to maximise log likelihood
- Number of free parameters (weights) scale
  - Linearly with vocabulary size
  - Linearly with context size
- Can be (linearly) interpolated with n-gram model
- Majority of the weights (hence majority of the computation) is in the output layer
- Reduce computation by only including the s most frequent words at the output — the shortlist (S) (full vocabulary still used for context)
- Use an n-gram model to estimate probabilities of words not in the shortlist

A Neural Probabilistic Language Model



Automatically Processing Tweets from Gang-Involved Youth: Towards Detecting Loss and Aggression

Case of Chicago: In 2015, 40% increase in firearm violence someone is shot every 2-3 hours in the city. The PD believes this might be because of the prevalence of “digital street”.

Dataset: 27,000 tweets posted by a young and particularly powerful female Chicago gang member, Gakirah Barnes, and people with whom she communicated.

Goal: The three categories are loss, retribution and other. Alert community outreach groups before grief turns to retribution.

Team: consists of social workers who identified the tweets and computer scientists who built the classification model in close collaboration. Chicago youth member helped them understand the gang-related dialect.

Non-standard english: use of emojis, use of dialectal (African American Vernacular English or AAVE) grammar and vocabulary, gang-related slang, abbreviations.

Challenges: POS taggers are not accurate, Wiktionary is also not useful.

They accomplish three tasks in this paper:

1. Design a new corpus that is annotated with discourse intention based on a deep read of the corpus, as well as POS tags.
2. NLP resources for the sub-language used by Chicago gang members, specifically a POS tagger and a glossary.
3. A system to identify the emotion conveyed by tweets, using the Dictionary of Affect in Language
    - Features for the emotion classifier are POS tags produced by the tagger in (2) and quantitative scores representing the affect of words.

Automatically Processing Tweets from Gang-Involved Youth: Towards Detecting Loss and Aggression

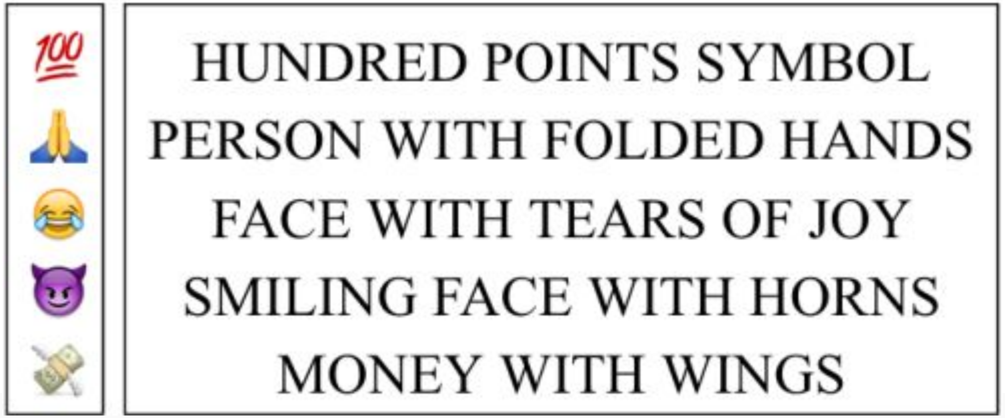
Problems with POS tagging: “lame ass Lil n*ggas”: The first annotator interpreted “ass” as an intensifying modifier to the adjective lame and tagged it is an adverb, while the second annotator read it as the second in a string of three adjectives modifying the n-word.

- Domain adaptation by adding a standardized tweet library

Cascading classifier that used two SVMs - first to group into “other” and “relevant” and second to group “relevant” into “grief” and loss”

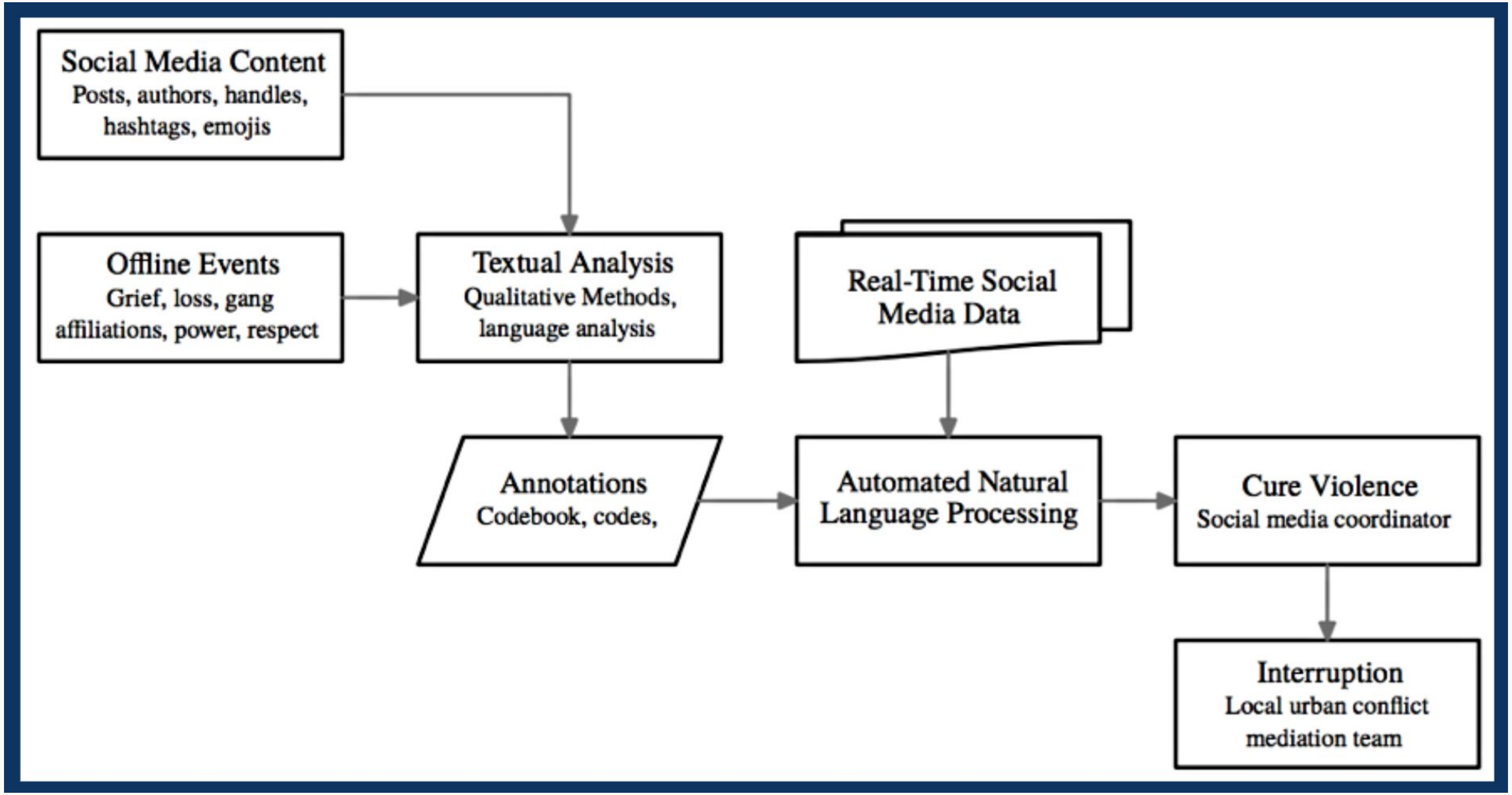
- Why are SVMs so good for NLP tasks?

Result: The classifier recognize loss tweets with 62.3% accuracy and aggression tweets with 63.6% accuracy, improving over the baseline by 13.7 points (aggression) and 5.8 points (loss). The CC classifier can identify both with 81.5% accuracy.



💯	HUNDRED POINTS SYMBOL
🙏	PERSON WITH FOLDED HANDS
😂	FACE WITH TEARS OF JOY
😈	SMILING FACE WITH HORNS
💸	MONEY WITH WINGS

Digital Urban Violence Analysis Approach: what triggered conversations to escalate into aggression


Social Media Content
Posts, authors, handles, hashtags, emojis
Offline Events
Grief, loss, gang affiliations, power, respect
Textual Analysis
Qualitative Methods, language analysis
Real-Time Social Media Data
Annotations
Codebook, codes,
Automated Natural Language Processing
Cure Violence
Social media coordinator
Interruption
Local urban conflict mediation team