In-Class Assignment: Topic Modeling

DATA 5420/6420

Name: Dallin Moore

Topic modeling is a more sophisticated approach to extracting topics/themes from a corpus than some of the more simple methods we discussed during class, like key phrase extraction. There are a number of algorithms that can be used to peform topic modeling, we will focus on the most common, called LDA.

- Latent Dirichlet Allocation

In this in-class assignment we will implement the above method on a corpus of research papers from the NeurIPS conference to try and extract meaningful topic labels for these papers. While we will focus on LDA, there are other methods for performing topic modeling like LSI, and NMF.

```
# load all dependencies
import nltk
import gensim

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

import re
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')

from sklearn.decomposition import LatentDirichletAllocation
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
import gensim
import matplotlib.pyplot as plt
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

1) Data Retrieval

We will be downloading the dataset directly using the following commands:

```
!wget https://cs.nyu.edu/~roweis/data/nips12raw_str602.tgz
```

```
--2024-03-08 03:49:06--  https://cs.nyu.edu/~roweis/data/nips12raw_str602.tgz
Resolving cs.nyu.edu (cs.nyu.edu)... 216.165.22.203
Connecting to cs.nyu.edu (cs.nyu.edu)|216.165.22.203|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12851423 (12M) [application/x-gzip]
Saving to: 'nips12raw_str602.tgz'

nips12raw_str602.tg 100%[===================>]  12.26M  20.5MB/s    in 0.6s

2024-03-08 03:49:07 (20.5 MB/s) - 'nips12raw_str602.tgz' saved [12851423/12851423]
```

A) Data Extraction

We need to uncompress this tgz file and extract the different text files within each of the subfolders, like so:

```
!tar -xzf nips12raw_str602.tgz
```

```
DATA_PATH = '/content/nipstxt'                                  # add in content path
print(os.listdir(DATA_PATH))
```

```
['README_yann', 'MATLAB_NOTES', 'nips11', 'nips06', 'nips10', 'nips05', 'idx', 'orig', 'nips00', 'nips07', 'nips09', 'nips03', 'nips01',
```

2) Basic Text Preprocessing

Let's think about which text cleaning step we want to implement here for a process like topic modeling, where the goal is to group together terms that tend to co-occur with one another across documents to essentially form clusters of terms -- which form topics

```
def preprocess_text(text):
    text = re.sub(r"[^a-zA-Z]", " ", text.lower())
    tokens = nltk.word_tokenize(text)
    tokens = [token for token in tokens if token not in stopwords.words('english')]
    lemmatizer = WordNetLemmatizer()
    lemmas = [lemmatizer.lemmatize(token) for token in tokens]

    return ' '.join(lemmas)
```

```
# specify the folders we want to extract from
folders = ["nips{0:02}".format(i) for i in range(0, 5)]
papers = []

for folder in folders: # create a list of filenames within each folder
    folder_path = os.path.join(DATA_PATH, folder)
    file_names = os.listdir(folder_path)

# for each file, create the full file path, read in the file, preprocess the text, append to list of all papers
for file_name in file_names:
    file_path = os.path.join(folder_path, file_name)
    with open(file_path, encoding='utf-8', errors='ignore') as f:
        data = f.read()
        preprocessed_data = preprocess_text(data)
        papers.append(preprocessed_data)
```

```
papers[0][:1000]
```

```
'node splitting constructive algorithm feed forward neural network mike wynne jones res
earch initiative pattern recognition st andrew road great malvern wr p uk mikewj hermes
mod uk abstract constructive algorithm proposed feed forward neural network us node spl
itting hidden layer build large network smaller one small network form approximate mode
l set training data split creates larger powerful network initialised approximate solut
ion already found insufficiency smaller network modelling system generated data lead os
cillation hidden node whose weight vector cover gions input space detail required model
node identified split two using principal component analysis allowing new node cover tw
```

3) Feature Engineering

Before we perform any sort of vectorization, we're going to narrow down our pool of words to more common n-grams; specifically bigrams where the min_df = 2, and the max_df = 0.70, meaning...

```
tv = TfidfVectorizer(min_df=2, max_df=0.7, ngram_range=(2,2))
dtm = tv.fit_transform(papers)
vocabulary = np.array(tv.get_feature_names_out())
dtm.shape
```

```
(144, 16858)
```

```
pd.DataFrame(dtm.toarray(), columns=vocabulary, index=['Doc_'+str(i) for i in range(len(papers))])
```

	aaai press	ability distribution	ability generalize	ability generate	ability model	ability network	ability neural	ability proc
Doc_0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
Doc_139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_140	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_141	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_142	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Doc_143	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

144 rows × 16858 columns

Why might we use bigrams here in place of unigrams?

There a lot of concepts within the texts like 'maching learning' that lose meaning when they are considered as unigrams, which makes bigrams more useful for this scenario.

4) Topic Modeling with Latent Dirichlet Allocation (LDA)

Let's start by first just running a topic model and examining the top features

```
# Fit LDA Model
lda = LatentDirichletAllocation(n_components=10, random_state=42)
lda.fit(dtm)
```

```
LatentDirichletAllocation
LatentDirichletAllocation(random_state=42)
```

A) Print topics with top 10 words per topic

```
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print("Topic %d:" % (topic_idx))
        print(", ".join([feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1]]))
```

```
no_top_words = 10
display_topics(lda, vocabulary, no_top_words)
```

```
Topic 0:
test set, silicon retina, weight decay, memory based, et al, hidden unit, feature extractor, feature map, dendritic tree, sp sp
Topic 1:
winner take, model merging, horizontal cell, take network, motion energy, target location, training data, domain theory, limit cycle, er
Topic 2:
spinal cord, network model, model image, expert network, et al, change model, horn usher, muscle tension, gating network, cross validati
Topic 3:
hidden unit, uniform convergence, center location, recurrent network, context unit, pid controller, relative frequency, knowledge base,
Topic 4:
weight decay, mistake rate, empirical risk, auditory nerve, tone burst, feature unit, segmentation recognition, spatial frequency, firin
Topic 5:
hidden unit, training set, control point, rational function, rem sleep, degree approximation, training data, learning curve, evoked pote
Topic 6:
hidden unit, wind speed, neural net, reinforcement learning, hebbian rule, decision tree, sensory input, block diagram, xt xt, parsec ne
Topic 7:
hidden unit, recurrent network, ocular dominance, gating network, motor command, neural net, parameter setting, network chip, node funct
```

```
Topic 8:
kernel regression, anti hebbian, basis function, hidden neuron, view object, linear combination, rule extraction, domain knowledge, leas
Topic 9:
output space, vc dimension, et al, dendritic tree, time constant, second order, learning rate, synaptic efficacy, mar model, testing dat
```

But how do we know if this is a reasonable number of topics? What are some ways we could define that?

By looking at topic cogerence we can see what the similarity is betweeen all of the topics to know if any of the topics are somewhat uncommon. We could also look at other metrics like perplexity to get a similar understanding.

B) Examining Topic Coherence

Let's now get an idea of how our topic model is performing by computing the perplexity and the coherence scores (UMass).

```
# sci-kit learn does not support coherenc score so a different lda model must be brought in for the score
def custom_tokenizer(text):
    # create unigrams
    tokens = nltk.word_tokenize(text)
    tokens = [token for token in tokens if token not in stopwords.words('english')]
    # create bigrams
    bigrams = ["_".join(tokens[i:i+2]) for i in range(len(tokens)-1)]
    return bigrams

tokenized_texts = [custom_tokenizer(text) for text in papers]                          # create a list of bigram strings for each paper
```

Annoyingly sklearn doesn't do coherence scores so we have to convert our LDA model to a gensim (another NLP library in python) model

```
gensim_dict = Dictionary(tokenized_texts)
gensim_dict.filter_extremes(no_below=2, no_above=0.7) # we need to set this to the same filtering we did during our vectorization step
print(gensim_dict)
```

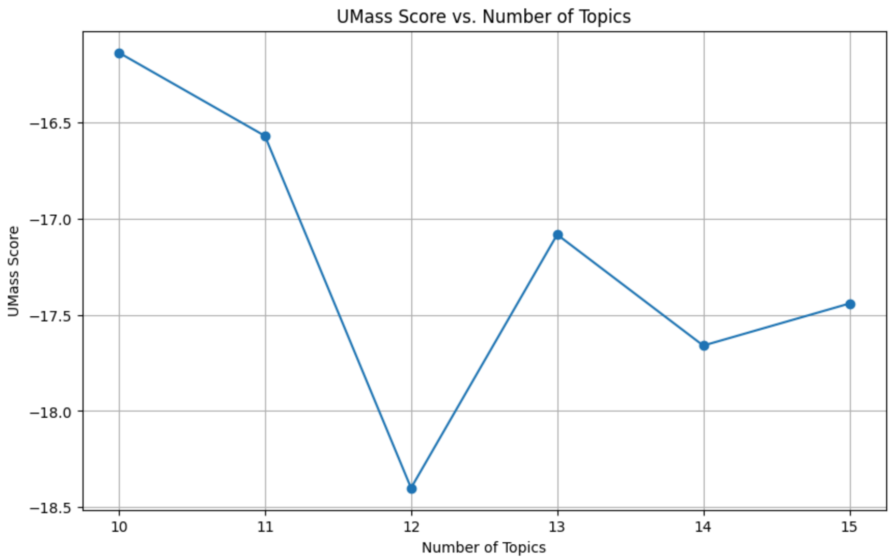
```
Dictionary<18643 unique tokens: ['ac_cording', 'acknowledgement_author', 'adaptive_network', 'add_new', 'added_weight']...>
```

Instead of just examining different numbers of topics individually, let's set a range of topics to examine coherence (UMass) scores for and then plot the change in coherence for each topic, we can do this efficiently with a for loop:

```
# Container to hold coherence values
coherence_values = []

topic_nums = range(10, 16) # try 10 to 15 topics

# fit LDA model for each value of topic_nums
for num_topics in topic_nums:
    lda_model = LatentDirichletAllocation(n_components=num_topics, random_state=42)
    lda_model.fit(dtm)
    # extract top features for each topic, replace with vocab word
    word_ids = lda_model.components_.argsort(axis=1)[:, ::-1][:, :10]
    topics = [[tv.get_feature_names_out()[i].replace(" ", "_") for i in topic_word_ids] for topic_word_ids in word_ids]
    # print extracted topics
    print(f"Extracted topics for num_topics={num_topics}:")
    for idx, topic in enumerate(topics):
        print(f"Topic {idx}: {topic}")
    # calculate and append the topic score
    coherence_model_lda = CoherenceModel(topics=topics, texts=tokenized_texts, dictionary=gensim_dict, coherence='u_mass')
    coherence_lda = coherence_model_lda.get_coherence()
    print(f"UMass for num_topics={num_topics}: {coherence_lda}\n")

    coherence_values.append(coherence_lda)
```

```
    Topic 2: ['hidden_unit', 'training_set', 'training_data', 'back_propagation', 'test_set', 'speech_recognition', 'data_set', 'hidden_
    Topic 3: ['rational_function', 'degree_approximation', 'xt_xt', 'spatial_frequency', 'synaptic_efficacy', 'two_population', 'impulse_
    Topic 4: ['gating_network', 'anti_hebbian', 'weight_decay', 'expert_network', 'vc_dimension', 'empirical_risk', 'risk_minimization', 
    Topic 5: ['ocular_dominance', 'parsec_network', 'auditory_nerve', 'tone_burst', 'basis_function', 'feature_extraction', 'projection_p
    Topic 6: ['output_space', 'second_order', 'knowledge_base', 'output_port', 'error_function', 'unit_pattern', 'feedback_controller', '
    Topic 7: ['vc_dimension', 'cross_validation', 'memory_based', 'motion_energy', 'input_window', 'neural_net', 'sensory_input', 'forwar
    Topic 8: ['intermediate_model', 'threshold_function', 'horn_usher', 'random_field', 'hough_transform', 'linear_threshold', 'markov_ra
    Topic 9: ['control_point', 'mistake_rate', 'uniform_convergence', 'time_constant', 'relative_frequency', 'th_neuron', 'optimal_path', 
    Topic 10: ['recurrent_network', 'hidden_neuron', 'horizontal_cell', 'motor_command', 'neural_net', 'pid_controller', 'ocular_dominanc
    Topic 11: ['spinal_cord', 'sample_size', 'projection_pursuit', 'parameter_setting', 'dendritic_tree', 'smooth_function', 'eye_positic
    Topic 12: ['feature_map', 'wind_speed', 'silicon_retina', 'second_order', 'gradient_descent', 'limit_cycle', 'new_term', 'sparse_poly
    UMass for num_topics=13: -17.08267913147392

    Extracted topics for num_topics=14:
    Topic 0: ['hidden_unit', 'et_al', 'training_set', 'neural_net', 'training_data', 'test_set', 'back_propagation', 'recurrent_network', 
    Topic 1: ['kernel_regression', 'prediction_risk', 'gating_network', 'center_location', 'cross_validation', 'pid_controller', 'tied_mi
    Topic 2: ['dendritic_tree', 'change_model', 'ocular_dominance', 'muscle_tension', 'synaptic_input', 'intermediate_model', 'inverse_ki
    Topic 3: ['rational_function', 'wind_speed', 'expert_network', 'model_merging', 'gating_network', 'degree_approximation', 'reinforcem
    Topic 4: ['ocular_dominance', 'evoked_potential', 'error_function', 'node_function', 'single_layer', 'feature_vector', 'random_field'
    Topic 5: ['parsec_network', 'phonetic_category', 'feature_unit', 'sp_sp', 'knowledge_base', 'occam_factor', 'speech_frame', 'speech_t
    Topic 6: ['winner_take', 'motor_command', 'segmentation_recognition', 'take_network', 'forward_dynamic', 'sensory_input', 'input_wind
    Topic 7: ['spinal_cord', 'mistake_rate', 'memory_based', 'output_port', 'feature_map', 'forward_model', 'darken_moody', 'visual_atten
    Topic 8: ['output_space', 'hidden_neuron', 'basis_function', 'horn_usher', 'learning_curve', 'term_memory', 'harmonic_function', 'sho
    Topic 9: ['uniform_convergence', 'relative_frequency', 'absolute_value', 'projection_pursuit', 'risk_minimization', 'linear_classifie
    Topic 10: ['network_chip', 'rem_sleep', 'horizontal_cell', 'self_organizing', 'vc_dimension', 'feature_extractor', 'limit_cycle', 'ey
    Topic 11: ['domain_theory', 'synaptic_efficacy', 'auto_adaptive', 'broad_phonetic', 'state_action', 'mar_model', 'testing_data', 'imp
    Topic 12: ['control_point', 'auditory_nerve', 'tone_burst', 'th_neuron', 'spatial_frequency', 'firing_rate', 'context_unit', 'reduced
    Topic 13: ['empirical_risk', 'view_object', 'linear_combination', 'silicon_retina', 'motion_energy', 'parameter_setting', 'model_imag
    UMass for num_topics=14: -17.65991778807524

    Extracted topics for num_topics=15:
    Topic 0: ['weight_decay', 'kernel_regression', 'recurrent_network', 'time_series', 'optimal_path', 'th_neuron', 'threshold_network', 
    Topic 1: ['sp_sp', 'inverse_kinematics', 'model_image', 'output_port', 'blind_separation', 'separation_source', 'viewing_position', '
    Topic 2: ['rational_function', 'degree_approximation', 'deterministic_annealing', 'temporal_difference', 'elastic_net', 'learning_tim
    Topic 3: ['anti_hebbian', 'memory_based', 'dendritic_tree', 'target_location', 'ocular_dominance', 'synaptic_competition', 'synaptic_
    Topic 4: ['empirical_risk', 'view_object', 'linear_combination', 'risk_minimization', 'confidence_interval', 'inner_product', 'linear
    Topic 5: ['projection_pursuit', 'hidden_unit', 'ocular_dominance', 'auditory_nerve', 'wind_speed', 'horizontal_cell', 'gating_network
    Topic 6: ['hidden_unit', 'training_set', 'et_al', 'neural_net', 'training_data', 'shown_figure', 'back_propagation', 'data_set', 'fig
    Topic 7: ['rem_sleep', 'mistake_rate', 'phonetic_category', 'feature_map', 'order_recurrent', 'reinforcement_learning', 'mamelak_hobs
    Topic 8: ['motion_energy', 'winner_take', 'parsec_network', 'model_merging', 'motor_command', 'take_network', 'second_order', 'forwar
    Topic 9: ['silicon_retina', 'sensory_input', 'context_unit', 'null_direction', 'absolute_value', 'risk_minimization', 'linear_classif
    Topic 10: ['eye_position', 'input_window', 'error_vector', 'error_correcting', 'visual_system', 'correcting_code', 'sensitive_neuron'
    Topic 11: ['spinal_cord', 'hidden_neuron', 'parameter_setting', 'synaptic_efficacy', 'impulse_response', 'constrained_optimization', 
    Topic 12: ['speech_recognition', 'output_space', 'markov_model', 'hidden_markov', 'spatial_frequency', 'continuous_speech', 'speech_s
    Topic 13: ['center_location', 'genetic_algorithm', 'basis_function', 'dendritic_tree', 'evoked_potential', 'self_organizing', 'featur
    Topic 14: ['control_point', 'uniform_convergence', 'expert_network', 'relative_frequency', 'change_model', 'network_chip', 'xt_xt', '
    UMass for num_topics=15: -17.441241931492744
```

```
# Plotting
plt.figure(figsize=(10, 6))
plt.plot(topic_nums, coherence_values, marker='o')
plt.title('UMass Score vs. Number of Topics')
plt.xlabel('Number of Topics')
plt.ylabel('UMass Score')
plt.grid(True)
plt.show()
```



How do we interpret U_Mass? What is the optimal number of topics between 10-15 for this dataset?

We want to maximize the coherence, so, for this dataset, the highest is 10 topics so that is the optimal number.

Rerun with optimal topic number

```
lda = LatentDirichletAllocation(n_components=10, random_state=42)
doc_topic = lda.fit(dtm)
```

Examine top features per topic

```
no_top_words = 10
display_topics(doc_topic, vocabulary, no_top_words)
```

```
Topic 0:
test set, silicon retina, weight decay, memory based, et al, hidden unit, feature extractor, feature map, dendritic tree, sp sp
Topic 1:
winner take, model merging, horizontal cell, take network, motion energy, target location, training data, domain theory, limit cycle, er
Topic 2:
spinal cord, network model, model image, expert network, et al, change model, horn usher, muscle tension, gating network, cross validati
Topic 3:
hidden unit, uniform convergence, center location, recurrent network, context unit, pid controller, relative frequency, knowledge base,
Topic 4:
weight decay, mistake rate, empirical risk, auditory nerve, tone burst, feature unit, segmentation recognition, spatial frequency, firin
Topic 5:
hidden unit, training set, control point, rational function, rem sleep, degree approximation, training data, learning curve, evoked pote
Topic 6:
hidden unit, wind speed, neural net, reinforcement learning, hebbian rule, decision tree, sensory input, block diagram, xt xt, parsec ne
Topic 7:
hidden unit, recurrent network, ocular dominance, gating network, motor command, neural net, parameter setting, network chip, node funct
Topic 8:
kernel regression, anti hebbian, basis function, hidden neuron, view object, linear combination, rule extraction, domain knowledge, leas
Topic 9:
output space, vc dimension, et al, dendritic tree, time constant, second order, learning rate, synaptic efficacy, mar model, testing dat
```

As mentioned before, we can augment our analysis here with ChataGPT, to help us understand these top features and apply a meaningful label.

Try copying and pasting the list above into chatgpt and asking it to provide topic labels for these neurips papers.

ChatGPT output:

- Topic 0: Neural Network Architecture
- Topic 1: Neural Network Training and Optimization
- Topic 2: Neural Network Models in Biomedical Research
- Topic 3: Neural Network Learning Algorithms
- Topic 4: Neural Network Performance Evaluation
- Topic 5: Neural Network Training Techniques
- Topic 6: Neural Network Applications in Wind Speed Prediction
- Topic 7: Neural Network Structures and Dynamics
- Topic 8: Neural Network Regression and Function Approximation
- Topic 9: Neural Network Model Evaluation and Testing

Finally, let's examine the topic breakdown by document

```
doc_topic_matrix = lda.transform(dtm)

df_doc_topic = pd.DataFrame(doc_topic_matrix, columns=[f'Topic {i}' for i in range(lda.n_components)])
df_doc_topic
```

1 to 25 of 144 entries Filter

index	Topic 0	Topic 1	Topic 2	Topic 3
131	0.004942673094920984	0.004942289577455942	0.004942880838545198	0.0049436443281216
61	0.005584664925841934	0.005584222416552015	0.005582485133957961	0.0055825195798130
4	0.0056703476832589686	0.00567555316228408	0.005672554718338546	0.0056711327664330
121	0.00580885092301035	0.005804717847846001	0.005804836233946573	0.0058047212335993
45	0.005840100156917262	0.005840108815880048	0.005840414636450893	0.00584018451429618
100	0.005992829883348409	0.0059925300559159666	0.005992220934979709	0.0059940144847840
24	0.006021780862420523	0.006022338174981959	0.006021961189475402	0.0060212768178432
125	0.0062769848732808385	0.006275872335958392	0.0062771746850215915	0.0062758454666082
108	0.006655478748239205	0.006655238221222403	0.006655049181825466	0.0066555105957150
133	0.006920723174368752	0.006919995781202914	0.006917564018910095	0.00691934157985292
95	0.006994988599614866	0.006994643134999802	0.00699649881377385	0.0069944516814710
75	0.007099075239637231	0.007097063189976745	0.007096239077848611	0.0071003238069600
3	0.007260584164659561	0.007264370149247903	0.00726045746665202	0.0072645845791020
86	0.007548238204585512	0.007549914842541046	0.0075460631146027374	0.0075452763593031
10	0.007774393180212528	0.007774290016497223	0.007775186118670513	0.00777485707223789
127	0.007976899949333663	0.007976797477905084	0.007981085115326539	0.0079770013793081
136	0.008489928914678582	0.008490504463411647	0.008500746817529433	0.0084893604261322
138	0.00927804225667408	0.00927250548695431	0.009272965602282466	0.0092726204565703
143	0.01242754923919575	0.01242404113025985	0.01242611614989674	0.012422186193699
88	0.01139330176023636	0.011393260590055831	0.011393688839808801	0.89744412639645
85	0.010543698482187738	0.9051214780747591	0.010543739525950696	0.0105411440573392
140	0.010514160076634584	0.01051451452183432	0.010517033611948562	0.0105137215049579
52	0.01040559402642464	0.010402295651128653	0.010405734222337355	0.0104013063856475
99	0.009954648825708451	0.009953064122156314	0.910428694048027	0.0099506090278754
30	0.009578737076397155	0.009578659931158175	0.009579543521770224	0.0095774997454961

Show 25 per page

1 2 3 4 5 6

Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: View recommended plots

```
papers[131][300:2000]
```

'problem number important practical issue identified discussed general theoretical per spective practical issue examined context case s
tudy td applied learning game backgammon outcome self play apparently first application algorithm complex nontrivial task found zero kn
owledge built network able learn scratch play entire game fairly strong intermediate level performance clearly better conventional comm
ercial program fact surpasses comparable network trained massive human expert data set hidden unit network apparently discovered useful
feature longstanding goal computer game research furthermore set hand crafted feature added input representation resulting network reac
h near expert level performance achieved good result world class human play introduction consider prospect application td algorithm del