

Biomedical literature mining

GRAPH KERNEL-BASED LEARNING FOR GENE–GENE
INTERACTION EXTRACTION

Ai-Ru Hsieh and Chen-Yu Tsai

European Journal of Medical Research, 2024

Why Biomedical Text Mining?

- Exponential growth of biomedical literature (>30 million PubMed citations)
- **Challenge:** Extracting relevant gene-gene interaction (GGI) data for precision medicine
 - GGIs can help explain the heritability of human complex diseases
 - Genetic interactions can help with detecting gene functions and pathways
- **Goal:** Automating the extraction of GGIs to reduce time and manual effort

GGI Extraction: Problems

- Existing methods rely heavily on supervised learning, which is costly and time-consuming
 - Limited number of corpora for GGIs in genetics
 - Current focus is primarily on protein-protein interactions (PPIs), not GGIs
- ⇒ **Idea:** Develop a graph kernel-based machine learning approach for automating the extraction of gene-gene interactions

Distant Supervision

- **Definition:** Automatically label data by aligning text with a knowledge base
 - Knowledge base: KEGG PATHWAY database (for human diseases)
- Annotate sentences in PubMed abstracts based on KEGG
- **Output:** A large, labeled corpus for training the machine learning model

Distant Supervision Corpus

1. Filtered abstracts from PubMed (1946–2021) based on keywords
 - Corpus: ~130.000 abstracts
 2. Text preprocessing
 - Named Entity Recognition (NER) using PubTator to recognize genes in abstracts
 - Extracted sentences of gene–gene co-occurrence using natural language model of ScispaCy (Final Corpus: ~390.000 sentences)
 - Sentence category determination:
 - Obtained direct & indirect gene–gene interactions from KEGG
 - Listed all possible gene pairs of the sentences
 - Positive category: Gene pair has a GGI in KEGG (~10.000)
 - Negative category: No GGI (~380.000)
- ⇒ Each data in the corpus contained: sentence with two genes, article source of the sentence, NCBI ID of the genes, and whether there is an interaction in KEGG.

Graph Feature Construction

1. Tokenization

2. Part-of-speech tagging (POS)




3. Dependency parsing

- Identify relationship between words in sentences & obtain the dependency relation between words

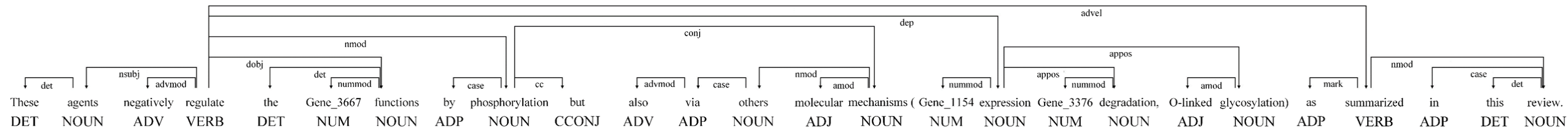
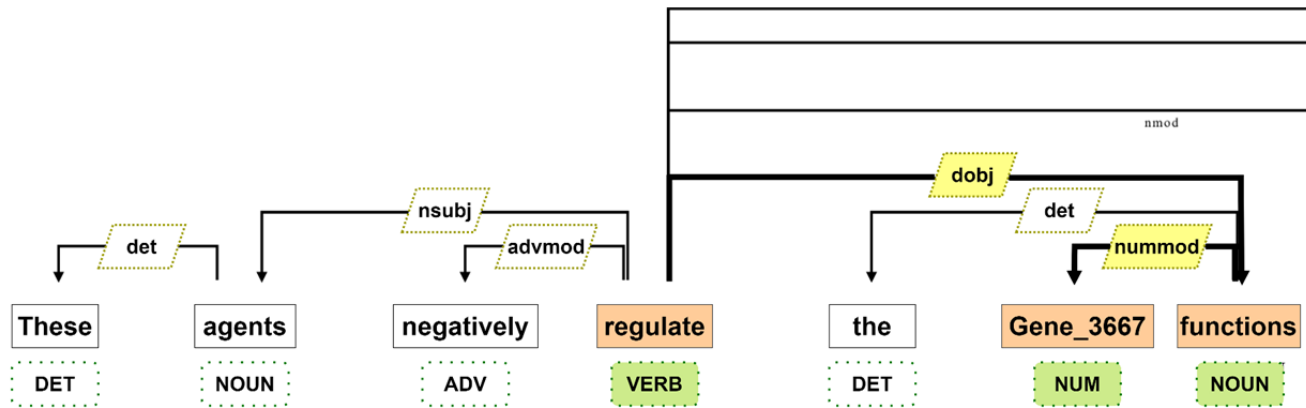
Graph-based Features

- Weighted graph for each sample/sentence of the corpus (by NetworkX)
 - **Vertices:** tokens in a sentence, with POS labels
 - **Edges:** dependency relations between tokens, with attributes
 - Attributes: Dependency labels, directional relationship
 - Shortest dependency path (SDP) of two entities contains the most information in a sentence
 - Edge belonging to the SDP was given a weight of 0.9, while the remaining were given a weight of 0.3
- ⇒ 4 features for each sentence sample to construct the NetworkX graphic data: POS, dependency relation, directional relationship, and SDP

Feature Graph (Example Sentence)

Dependency relation 
 Text 
 POS tags 

advmod: adverbial modifier
amod: adjectival modifier
appos: appositional modifier
case: case marking
cc: coordination
conj: conjunct
dep: dependent
det: determiner
dobj: direct object
mark: marker, such as "that", "whether", "because" and "when"
nmod: nominal modifier
nummod: numeric modifier
num: numeric modifier



SVM Graph Kernel

- Feature graphs were classified by the Support Vector Machine (SVM) graph kernel method to extract relationships
- 3 Kernel methods (Graph Kernel Functions):
 - Shortest Path (SPK)
 - Weisfeiler–Lehman Subtree (WLK)
 - Neighborhood Hash (NHK)
- **Definition Graph Kernel:** Input: Graphs, Output: Similarity of graph pairs

Graph Kernel Functions

Shortest Path Kernel (SPK)

- Calculates similarity between two graphs by comparing the shortest paths (SDP) between vertices
 - Focuses on path-based relationships between vertices
1. Shortest path between all pairs of vertices is computed
 2. Measures the similarity between two graphs by comparing the shortest paths:
 - Paths are matched based on their lengths and attributes
 - If two paths have similar properties (edge sequence), they contribute positive to the kernel similarity score

Graph Kernel Functions

Weisfeiler–Lehman Kernel (WLK)

- Compares structure of graphs
- Effective for capturing hierarchical & structural features of graphs
- **Label Refinement:**
 - Vertex labels are updated based on:
 - Current label
 - Neighboring vertex labels
- After multiple iterations, the kernel measures graph similarity:
 - Compares frequency of refined labels across graphs
 - If two graphs have similar label distributions, they are considered similar

Graph Kernel Functions

Neighborhood Hash Kernel (NHK)

- Compares local neighborhood structures around vertices
 - **Definition Neighborhood:**
 - Immediate set of connected vertices
 - Local neighborhood structure includes the vertex' direct relationships to its surrounding vertices
 - **Feature Hashing:**
 - Hashes local neighborhood features to create compact representations
 - Each hash serves as a unique identifier for a specific neighborhood structure
 - Compares graphs by analyzing similarities in hashed neighborhood structures
 - If two graphs have many similar neighborhood hashes, their similarity score increases
- ⇒ **Output of all Kernel functions:** Similarity matrix of graphs, where each value represents how similar a feature graph is to another.

SVM Classifier

Training

- SVM is trained with automatically labeled graph features
 - During training, SVM learns an optimal separating line (hyperplane) to classify new sentences as positive (GGI) or negative (no GGI)
 - **Goal:** correctly categorize new, previously unknown sentences into positive or negative
 - **Input:** Feature Graphs → Graph Kernel Matrix (SPK, WLK, NHK) & sentence category labels
 - **Output:** Optimal hyperplane to separate positive and negative sentences.
- ⇒ By optimising a hyperplane, the SVM learns to classify sentences as positive (GGI) or negative (no GGI) based on the kernel similarity scores.

SVM Classifier

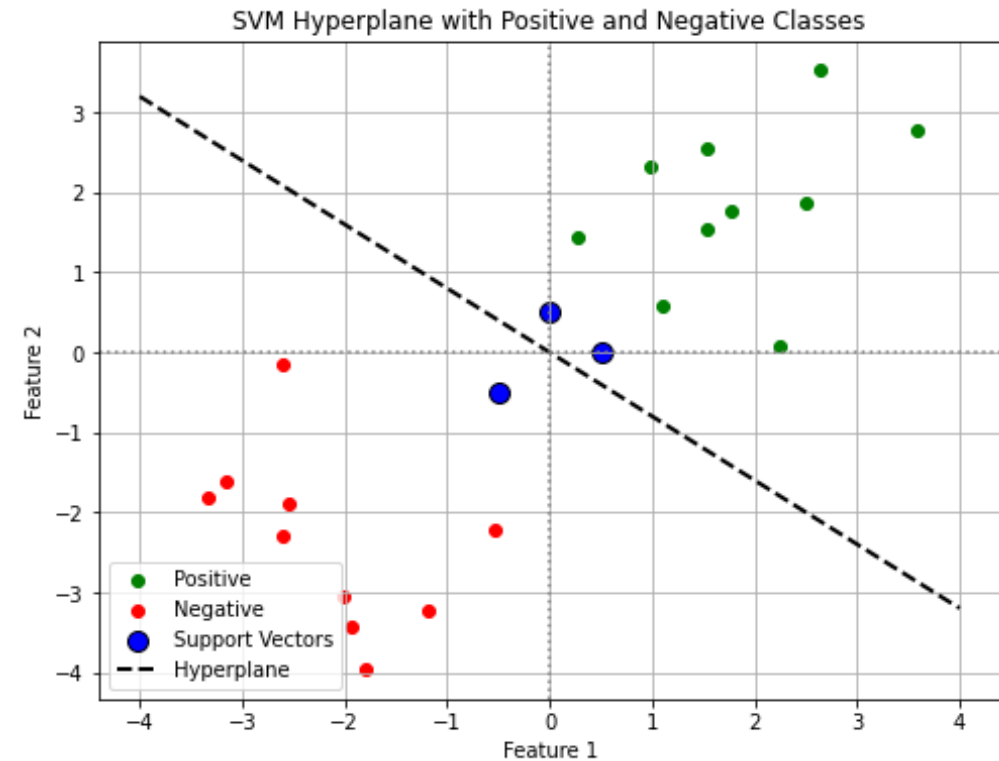
Training: Hyperplane

1. High-dimensional space:

- SVM projects the similarity vectors into a high-dimensional space
- Each graph (sentence) is a point in this space
- Similarities from the graph kernel function determine how “close” two points in the space are to each other

2. Optimal Hyperplane:

- SVM searches for the Hyperplane (dividing line) that best separates the classes (positive vs. negative)
- Goal: Maximize the distance (margin) between the hyperplane and the nearest data points



SVM Classifier

Prediction

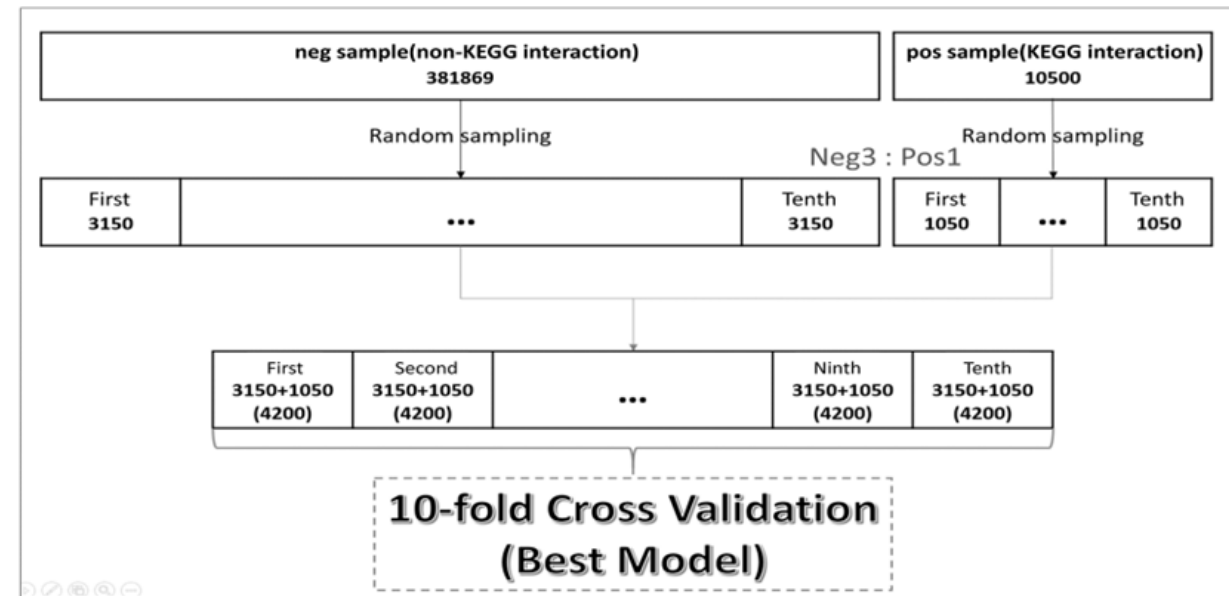
1. Construct feature graph of new sentence
 2. Calculate similarity vector between the new graph and all graphs from the training data set using the graph kernel function (SPK, WLK, NHK)
 3. Input similarity vector in trained SVM classifier
 4. SVM projects new sentence in high-dimensional space learned during training
 5. Based on position of the new sentence relative to the hyperplane, the SVM predicts:
 - **Positive:** The sentence describes a GGI
 - **Negative:** The sentence does not describe an interaction
- How reliable the prediction is, is based on the distance of the point to the hyperplane

Input: Graph representations of sentences

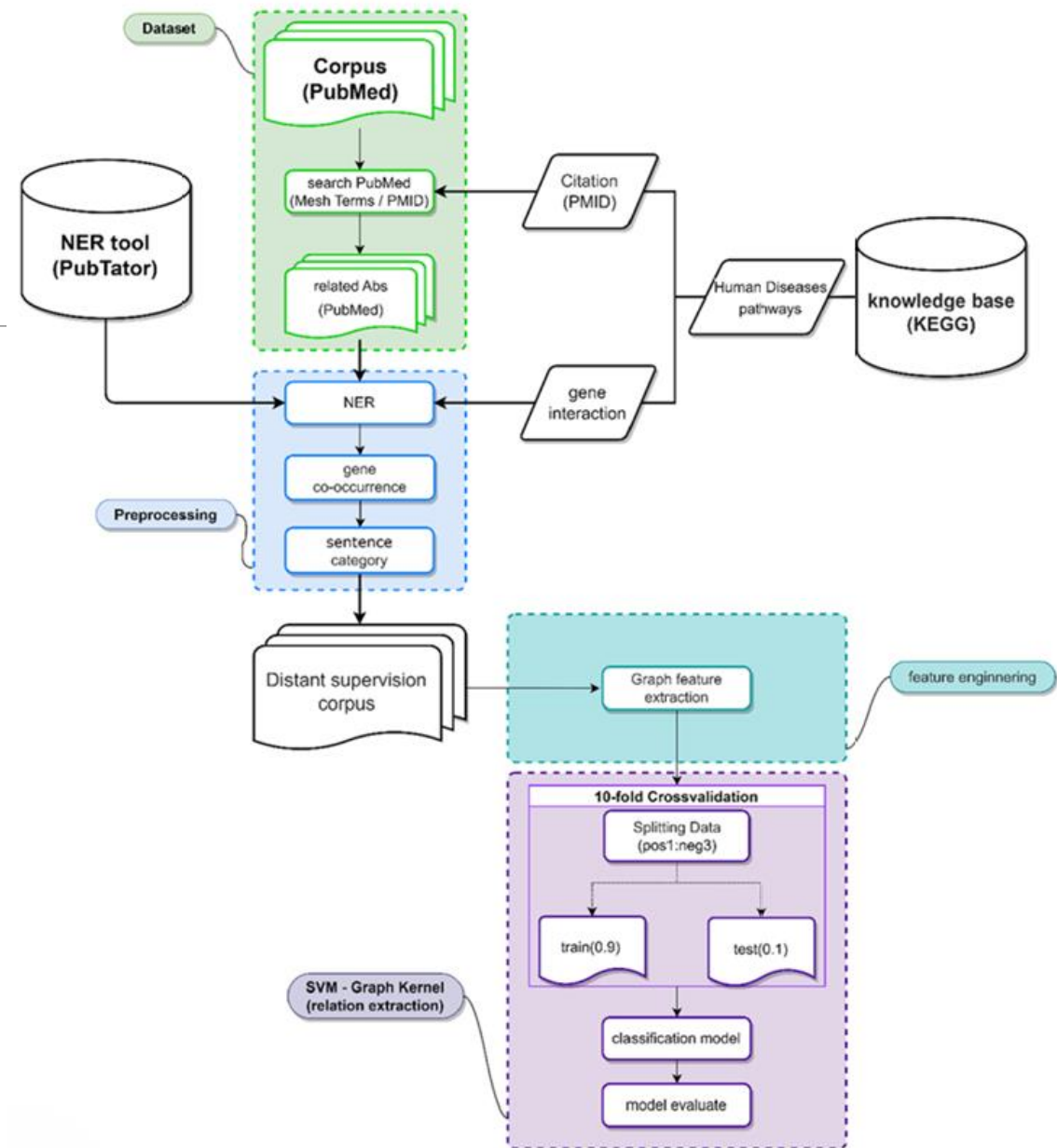
Output: Prediction (positive/negative) whether a gene-gene interaction is present

10-fold Cross Validation (10-fold CV)

- To avoid overfitting
- Big difference in positive (~10.000) and negative (~380.000) ratio (1:40)
→ random sampling → new ratio (1:3)
- **Final dataset** was divided into tenfold, each with 4,200 samples
- **Training set:** 9/10
Test set: 1/10



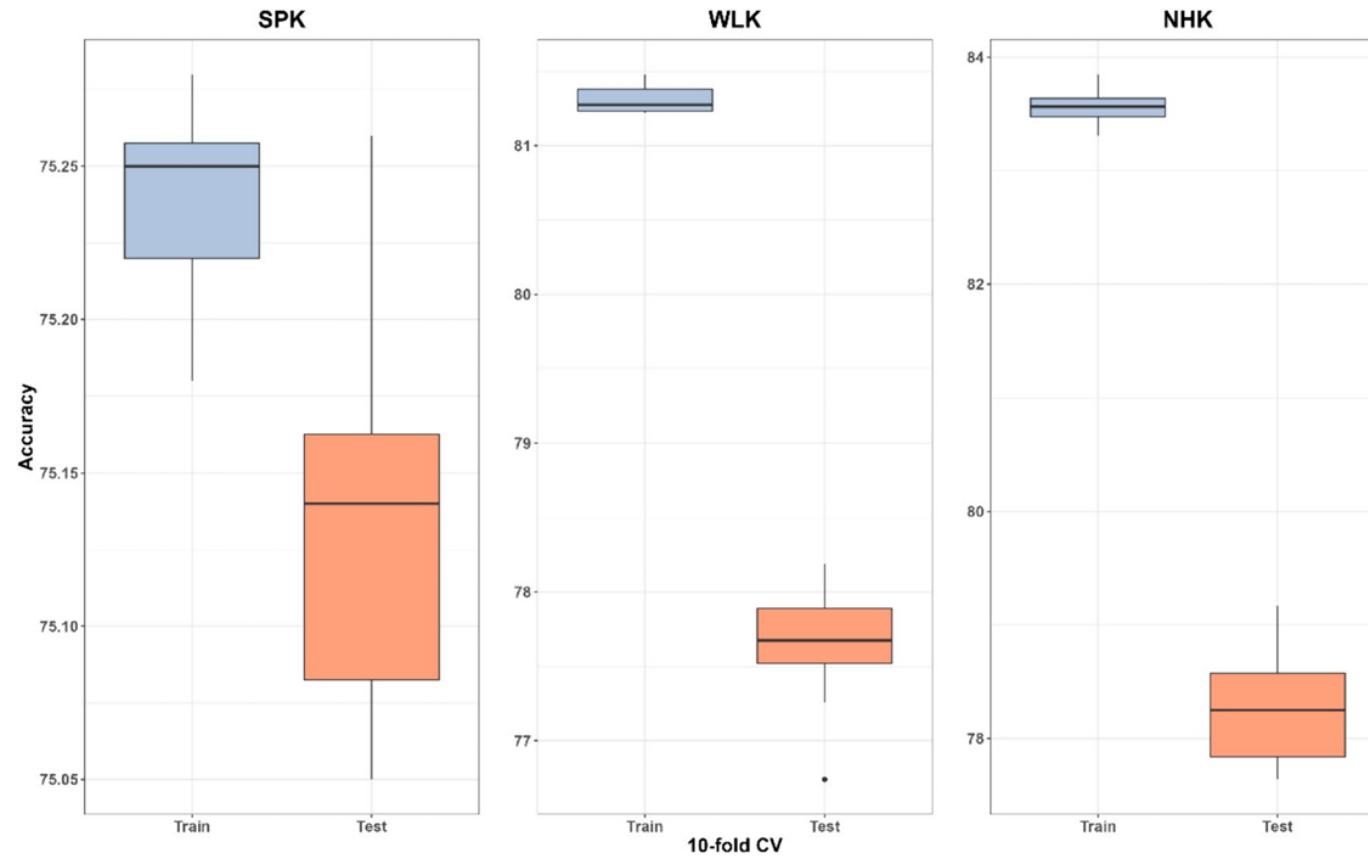
Overview



Flow diagram showing the process of analysis for this study

Results

10-fold CV



The results of tenfold CV using three graph kernels methods in shortest path kernel (SPK), Weisfeiler–Lehman kernel, (WLK) and neighborhood hash kernel (NHK). The accuracy of training data with (blue) and testing data (orange) is shown

Results

Classification

Classification results from SPK, WLK, and NHK

Graph kernel	Accuracy	Precision	Recall	F-score
Shortest path	0.75	0.81	0.75	0.75
Weisfeiler–Lehman	0.78	0.78	0.78	0.78
Neighborhood hash	0.79	0.79	0.79	0.79

Discussion

- **Strengths:**
 - Automated labeling via distant supervision reduces annotation cost
 - Graph kernels effectively capture sentence structure and meaning
 - **Limitations:**
 - Reliance on KEGG PATHWAY limits detection to known GGIs
 - Corpus of GGI is very scarce (1:3 ratio) → 10-fold CV to increase the stability
 - Limited generalizability to unknown interaction types
 - **Future Work:**
 - Incorporate additional knowledge bases (e.g. Gene Ontology)
 - Explore deep learning models for better feature extraction and classification
- ⇒ Alternative method to GGI text extraction

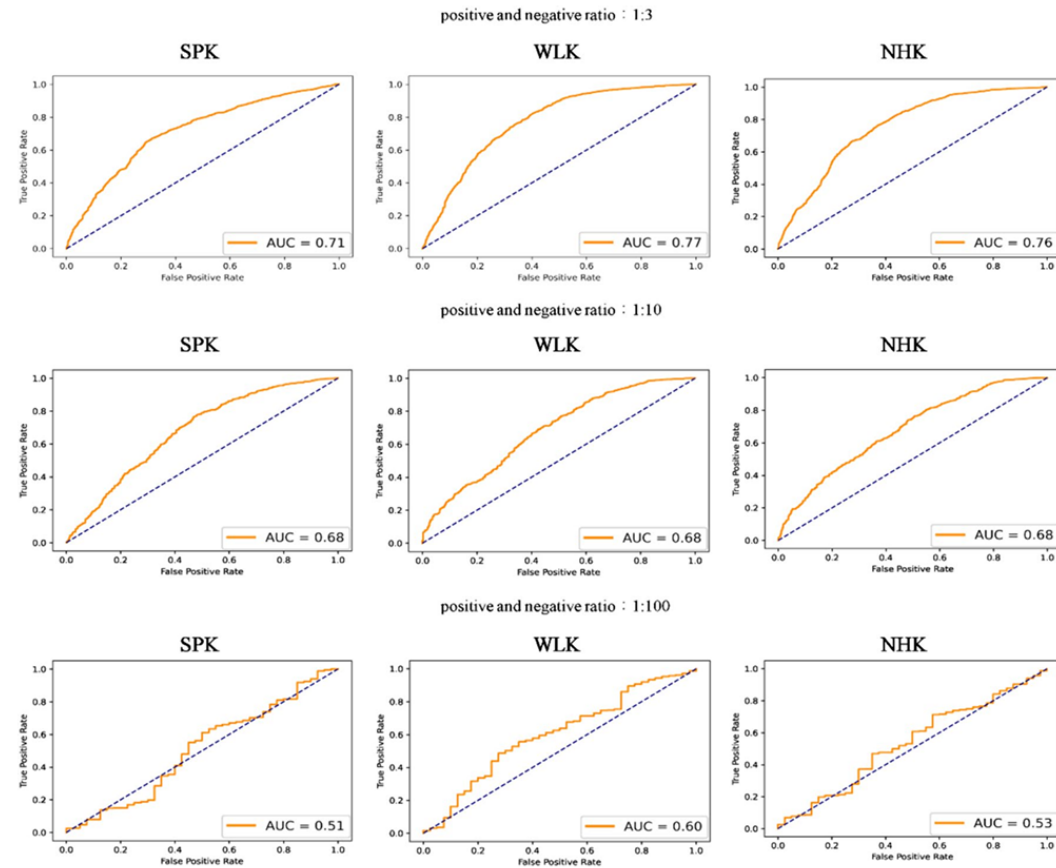
Summary

- Developed a novel distant supervision approach for automatic corpus creation (reduces manual effort)
- Leveraged a graph kernel machine learning technique to identify and extract GGIs from textual data
- Provided a robust and accurate tool for researchers in genomics
- Increased automation and accuracy in identifying GGIs to help understand human diseases

Thank you for listening!

Results

Classification



Receiver-operating characteristic curves for the three graph kernel methods in shortest path kernel (SPK), Weisfeiler–Lehman kernel (WLK), and neighborhood hash kernel (NHK) on the three positive and negative ratios (1:3, 1:10, and 1:100)

Graph Kernel Functions

Shortest Path Kernel (SPK):

- Compares shortest paths between vertices in graphs
- Focuses on path-based relationships between nodes

Weisfeiler–Lehman Kernel (WLK):

- Iteratively refines vertex labels and compares graph structures
- measures the structural similarity between graphs

Neighborhood Hash Kernel (NHK):

- Compares local neighborhood structures of vertices
- Focuses on subgraph similarity and small-scale graph features

⇒ **Output:** Similarity matrix