語の類似性判定技術を用いた
探索的テストの効率を向上させる手法の提案

電気通信大学　情報理工学部
西康晴研究室
柴崎　友輔

探索的テストとはどのようなテストか

- スマートスピーカーであるLINE Clovaの探索的テスト

本研究での探索的テストの定義

- **直前までに行われたテスト結果をもとにテスト対象システムについて学習し、テストエンジニアの経験則や直感といったセンスを用いることで質の高いテストケースを設計するテストアプローチ**

探索的テストには4つの重要なポイント

- 気づき　　：直前までに行われたテストによって得られた
  テスト対象の振る舞いやバグの情報から、
  何かしらの情報を獲得すること
- 学習　　：テストエンジニアがテスト対象の側面を
  何かしら獲得すること
- テスト設計：気づきや学習を元に、
  センスを活かしてテストを動的に設計すること
- テスト実行：テストを実行していくこと

Boosted Exploratory Testing

- 4つの重要なポイント(気づき・学習・テスト設計・テスト実行)のいずれか、または複数をツールを用いてブーストすることで探索的テストのバグ検出数やスピードを向上させる
  - この考え方をBoosted Exploratory Testing(BET)と呼ぶ
  - 自動化によって人間を排除するというコンセプトではない

テストエンジニアがツールの補助によって
より良い結果を得られるようにする

テスト設計をツールによってブーストする

- 本研究ではテストエンジニアのセンスをツールでブーストすることで、
  よりよくテストを設計する
  - センスとは、テストエンジニアの経験則や直感、閃きや
    違和感を感じとる洞察力などのことである
  - テストエンジニアは豊かなセンスを用いて
    バグを検出できる確率の高いテストケースを思いついていく

テストエンジニアのセンスをブーストする

- 本研究では２つのアプローチによって、ツールにセンスをブーストさせる

1. テストエンジニアがセンスを用いて思いついたテストケースをもとに、
   ツールにバグを検出する確率の高いテストケースを提示させる

2. ツールが提示させたテストケースをヒントとして用いることで、
   テストエンジニアにバグを検出する確率の高いテストケースを思いつかせる

共通の特徴をもったバグを芋づる式に見つけていく

- 本研究では「バグは偏在する」という法則に着目して
  バグを検出する確率の高いテストケースを
  ツールが提示したりテストエンジニアが思いつくことで
  バグを芋づる式に見つけていく

  - 異なるバグであっても、
    ある特定のバグのペアやグループは何か共通の特徴を持っていることが多い

  - バグA－（AとBの共通の特徴α）→
    バグB－（BとCの共通の特徴β）→
    バグC－（CとDの共通の特徴γ）→ バグD…
    と探索していくと、芋づる式にどんどんバグが見つかっていくのではないか？
    - 例：―宮町を別の地域と誤認識する―(似た読みの地域が複数存在する市区町村)→
      美里町では天気予報機能が動作しない
    - 特徴の少ないテストケースでは探索がすぐに終わってしまってバグが見つからない

自然言語によるテストデータは様々な特徴を持つ

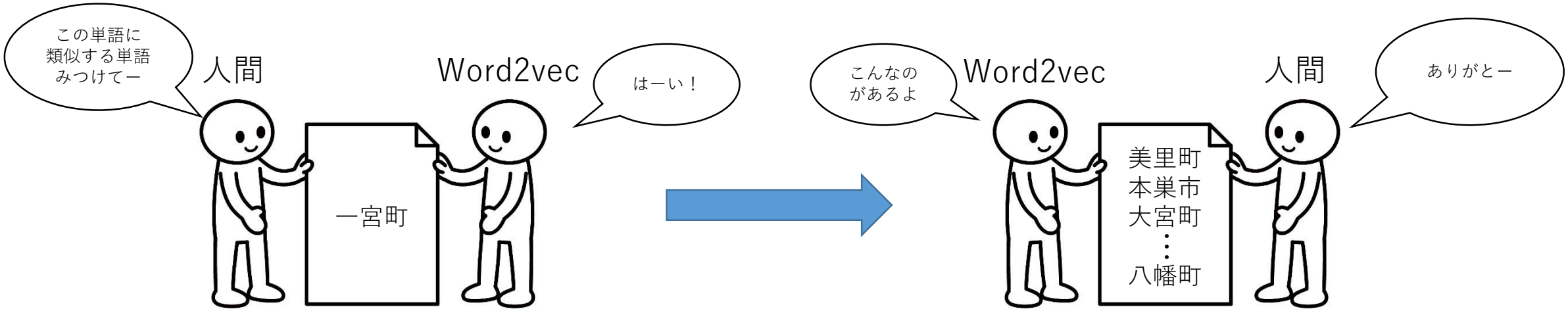
- 本研究ではさまざまな特徴を持つテストケースとして
  自然言語をテストデータとして持つテストケースを対象とする
  - 同音異義語がある、異音同義語がある、カナカナが入っている
    当て字がある、動的に意味の変化する文字（々）がある、など
  - 自然言語で音声操作を行うスマートスピーカーなどが対象となる

自然言語によるテストデータは様々な特徴を持つ

- さまざまな特徴を持つテストケースでも
  実際に人間がバグを発生させる可能性のある特徴を
  見つけていくことは難しい
  - 「一宮町の天気は？」というテストケースを実行したら
    「一宮市の天気は～です」とClovaが遠く離れた別の地域の天気情報を出力してしまう
    このような情報だけで多くの特徴はなかなか見つけだせない

ツールによる共通の特徴を持ったテストケースの提示

- 共通の特徴をツールに見つけさせるために、
  ニューラルネットワークを基にした自然言語処理技術を用いる
  - ニューラルネットワークは判断の理由が分かりにくいという性質を逆手にとって、
    人間には思いつかないような特徴を見つけさせることができる
  - Word2vecという語の類似性判定技術を用いてツールを実装する
    - Word2vecは単語を入力することで共通の特徴を持つ単語を出力することができる



ツールによる共通の特徴を持ったテストケースの提示

- Word2vecは入力単語と共通の特徴をもった単語を生成することができる
  - Word2vecはSkip-Gramというニューラルネットワークのモデルがもとになっている
  - Word2vecは語の意味のベクトル表現化している
  - コサイン類似度を用いることで入力語と類似する分布特性をもった単語を抽出している
  - 類似する分布特性を持った単語同士は何かしらの共通の特徴をもつと考えられる

入力単語と共通の特徴をもった単語を生成するという性質を用いて、
入力テストケースと共通の特徴を持ったテストケースを提示させる

Word2vecによって提示されたテストケースの実例

- 「一宮町」を入力することで提示された実際のテストケース列
- テストケース列をヒントにバグ検出テストケースの特徴をテストエンジニアがみつけていく

南部町の天気は？
御津町の天気は？
有明町の天気は？
本巣市の天気は？
入来町の天気は？
愛知郡の天気は？
石川郡の天気は？
那賀郡の天気は？
和気郡の天気は？
久米郡の天気は？
大分郡の天気は？
佐波郡の天気は？

豊田郡の天気は？
田川郡の天気は？
神崎郡の天気は？
三原郡の天気は？
山本郡の天気は？
たつの市の天気は？
藤岡町の天気は？
楠町の天気は？
相生市の天気は？
山田町の天気は？
吉野川市の天気は？
弥富市の天気は？

庄原市の天気は？
福津市の天気は？
天草市の天気は？
東近江市の天気は？
橋本市の天気は？
山武市の天気は？
鹿島町の天気は？
落合町の天気は？
日吉村の天気は？
阿波市の天気は？
三橋町の天気は？
生野町の天気は？

山城町の天気は？
東郷町の天気は？
湯梨浜町の天気は？
美郷町の天気は？
東浦町の天気は？
志賀町の天気は？
明和町の天気は？
美里町の天気は？
与謝野町の天気は？
伊奈町の天気は？
大和村の天気は？
川西町の天気は？

三島郡の天気は？
麻植郡の天気は？
大宮町の天気は？
清水町の天気は？
八幡町の天気は？
北方町の天気は？
松田町の天気は？
春日町の天気は？
朝日町の天気は？
東栄町の天気は？
大和町の天気は？
日高郡の天気は？

紀の川市の天気は？
瑞穂市の天気は？
赤磐市の天気は？
古志郡の天気は？
高田村の天気は？
大野村の天気は？
山口村の天気は？
坂本村の天気は？
佐伯郡の天気は？
南埼玉郡の天気は？
白浜町の天気は？
伊豆の国市の天気は？

Word2vecによって提示されたテストケースの実例

- 「一宮町」を入力することで提示された実際のテストケース列
- どのような特徴が目につきますか？

南部町の天気は？
御津町の天気は？
有明町の天気は？
本巣市の天気は？
入来町の天気は？
愛知郡の天気は？
石川郡の天気は？
那賀郡の天気は？
和気郡の天気は？
久米郡の天気は？
大分郡の天気は？
佐波郡の天気は？

豊田郡の天気は？
田川郡の天気は？
神崎郡の天気は？
三原郡の天気は？
山本郡の天気は？
たつの市の天気は？
藤岡町の天気は？
楠町の天気は？
相生市の天気は？
山田町の天気は？
吉野川市の天気は？
弥富市の天気は？

庄原市の天気は？
福津市の天気は？
天草市の天気は？
東近江市の天気は？
橋本市の天気は？
山武市の天気は？
鹿島町の天気は？
落合町の天気は？
日吉村の天気は？
阿波市の天気は？
三橋町の天気は？
生野町の天気は？

山城町の天気は？
東郷町の天気は？
湯梨浜町の天気は？
美郷町の天気は？
東浦町の天気は？
志賀町の天気は？
明和町の天気は？
美里町の天気は？
与謝野町の天気は？
伊奈町の天気は？
大和村の天気は？
川西町の天気は？

三島郡の天気は？
麻植郡の天気は？
大宮町の天気は？
清水町の天気は？
八幡町の天気は？
北方町の天気は？
松田町の天気は？
春日町の天気は？
朝日町の天気は？
東栄町の天気は？
大和町の天気は？
日高郡の天気は？

紀の川市の天気は？
瑞穂市の天気は？
赤磐市の天気は？
古志郡の天気は？
高田村の天気は？
大野村の天気は？
山口村の天気は？
坂本村の天気は？
佐伯郡の天気は？
南埼玉郡の天気は？
白浜町の天気は？
伊豆の国市の天気は？

Word2vecによって提示されたテストケースの実例

- 「一宮町」を入力することで提示された実際のテストケース列
- どのような特徴が目につきますか？

南部町の天気は？
御津町の天気は？
有明町の天気は？
本巣市の天気は？
入来町の天気は？
愛知郡の天気は？
石川郡の天気は？
那賀郡の天気は？
和気郡の天気は？
久米郡の天気は？
大分郡の天気は？
佐波郡の天気は？

豊田郡の天気は？
田川郡の天気は？
神崎郡の天気は？
三原郡の天気は？
山本郡の天気は？
たつの市の天気は？
藤岡町の天気は？
楠町の天気は？
相生市の天気は？
山田町の天気は？
吉野川市の天気は？
弥富市の天気は？

庄原市の天気は？
福津市の天気は？
天草市の天気は？
東近江市の天気は？
橋本市の天気は？
山武市の天気は？
鹿島町の天気は？
落合町の天気は？
日吉村の天気は？
阿波市の天気は？
三橋町の天気は？
生野町の天気は？

山城町の天気は？
東郷町の天気は？
湯梨浜町の天気は？
美郷町の天気は？
東浦町の天気は？
志賀町の天気は？
明和町の天気は？
美里町の天気は？
与謝野町の天気は？
伊奈町の天気は？
大和村の天気は？
川西町の天気は？

三島郡の天気は？
麻植郡の天気は？
大宮町の天気は？
清水町の天気は？
八幡町の天気は？
北方町の天気は？
松田町の天気は？
春日町の天気は？
朝日町の天気は？
東栄町の天気は？
大和町の天気は？
日高郡の天気は？

紀の川市の天気は？
瑞穂市の天気は？
赤磐市の天気は？
古志郡の天気は？
高田村の天気は？
大野村の天気は？
山口村の天気は？
坂本村の天気は？
佐伯郡の天気は？
南埼玉郡の天気は？
白浜町の天気は？
伊豆の国市の天気は？

Word2vecによって提示されたテストケースの実例

- 「一宮町」を入力することで提示された実際のテストケース列
- どのような特徴が目につきますか？

南部町の天気は？
御津町の天気は？
有明町の天気は？
本巣市の天気は？
入来町の天気は？
愛知郡の天気は？
石川郡の天気は？
那賀郡の天気は？
和気郡の天気は？
久米郡の天気は？
大分郡の天気は？
佐波郡の天気は？

豊田郡の天気は？
田川郡の天気は？
神崎郡の天気は？
三原郡の天気は？
山本郡の天気は？
たつの市の天気は？
藤岡町の天気は？
楠町の天気は？
相生市の天気は？
山田町の天気は？
吉野川市の天気は？
弥富市の天気は？

庄原市の天気は？
福津市の天気は？
天草市の天気は？
東近江市の天気は？
橋本市の天気は？
山武市の天気は？
鹿島町の天気は？
落合町の天気は？
日吉村の天気は？
阿波市の天気は？
三橋町の天気は？
生野町の天気は？

山城町の天気は？
東郷町の天気は？
湯梨浜町の天気は？
美郷町の天気は？
東浦町の天気は？
志賀町の天気は？
明和町の天気は？
美里町の天気は？
与謝野町の天気は？
伊奈町の天気は？
大和村の天気は？
川西町の天気は？

三島郡の天気は？
麻植郡の天気は？
大宮町の天気は？
清水町の天気は？
八幡町の天気は？
北方町の天気は？
松田町の天気は？
春日町の天気は？
朝日町の天気は？
東栄町の天気は？
大和町の天気は？
日高郡の天気は？

紀の川市の天気は？
瑞穂市の天気は？
赤磐市の天気は？
古志郡の天気は？
高田村の天気は？
大野村の天気は？
山口村の天気は？
坂本村の天気は？
佐伯郡の天気は？
南埼玉郡の天気は？
白浜町の天気は？
伊豆の国市の天気は？

Word2vecによって提示されたテストケースの実例

- テストケース列をヒントにすることで、
  バグを発生させる原因となっている可能性のある
  特徴を見つけることができるようにする
  - 例：市区町村の中でも町ではバグ起こりそうなんじゃないかな？
    漢数字が入った市区町村って誤認識されそうじゃないかな？
    村などの小さな市区町村は天気予報機能が動作しないのではないか？

ツールの補助によってテストエンジニアの
閃きや直感といったセンスをブーストしている

Word2vecを用いた探索的テストの手順

- 本研究では探索的テストを以下のステップで行う
  1. テストエンジニアがテストケースを思いつき、実行する
  2. テストケースをWord2vecに入力して、
     類似テストケース列を提示させる
  3. テストケース列をどのように用いるかを以下から選択する
     3.1. 2.で提示されたテストケース列を
          そのまま候補テストケースとする
     3.2. 2.で提示されたテストケース列から
          テストエンジニアのセンスでバグが出そうなものだけを選び出し
          候補テストケースとする
     3.3. 2.で提示されたテストケース列をヒントにして、
          テストエンジニアが新たなテストケースを思いつき、
          候補テストケースとする
  4. 候補テストケースを実行する
  5. 以下のいずれかをテストエンジニアのセンスで選ぶ
     5.1. バグが検出され、セッション終了時刻前であれば、2.に戻る
     5.2. バグが検出されず、セッション終了前であれば、1.に戻る
     5.3. バグが検出されず、セッション終了時刻が来たのであれば、セッションを終了する

検証

- 検証目的
  - ツールを用いて探索的テストを行うことで
    バグの検出数を増加させることができたのかどうかを確かめる

検証

- 検証方法
  - テスト対象　　：LINE Clova[8]の天気予報機能
  - 被験者　　：探索的テストの経験のない大学生6名
    3名ずつ2グループに分けてツールを用いた探索的テストと
    ツールを用いなかった探索的テストをそれぞれ1時間行わせる
  - 共通で与えるもの：チャーター、探索的テストの説明、バグを検出したテストケース、
    バグの偏在性の説明、テスト対象機能の利用方法
  - 評価基準　　：単位時間あたりのバグ検出数が増加したかどうか

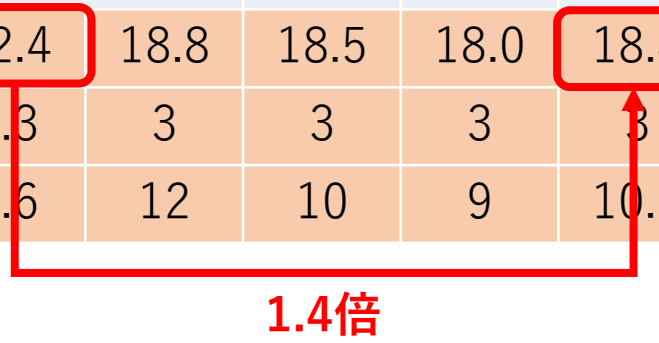
検証

- 検証結果

  ツールを用いた探索的テストはツールを用いなかった探索的テストに比べて
  バグの検出率(バグの総検出数/全テストケース生成数)は1.4倍に増加した

表1.　比較結果

探索的テスト手法	ツール未使用				ツール使用			
被験者	A	B	C	平均	D	E	F	平均
全テストケース生成数	27	40	49	38.6	64	54	50	56
バグの総検出数	4	5	5	4.6	12	10	9	10.3
所要時間	1	1	1	1	1	1	1	1
バグの検出率	14.5	12.5	10.2	12.4	18.8	18.5	18.0	18.4
検出したバグの種類数	2	2	3	2.3	3	3	3	3
バグの検出効率	4	5	5	4.6	12	10	9	10.3



検証

- 検証結果

  ツールを用いた探索的テストはツールを用いなかった探索的テストに比べて
  テストケースの生成数は1.4倍に増加した

表1.　比較結果

探索的テスト手法	ツール未使用				ツール使用			
被験者	A	B	C	平均	D	E	F	平均
全テストケース生成数	27	40	49	38.6	64	54	50	56
バグの総検出数	4	5	5	4.6	12	10	9	10.3
所要時間	1	1	1	1	1	1	1	1
バグの検出率	14.5	12.5	10.2	12.4	18.8	18.5	18.0	18.4
検出したバグの種類数	2	2	3	2.3	3	3	3	3
バグの検出効率	4	5	5	4.6	12	10	9	10.3

検証

- 検証結果
  - ツールを用いた探索的テストはツールを用いなかった探索的テストに比べて検出したバグの種類数は1.3倍に増加した

表1.　比較結果

探索的テスト手法	ツール未使用				ツール使用			
被験者	A	B	C	平均	D	E	F	平均
全テストケース生成数	27	40	49	38.6	64	54	50	56
バグの総検出数	4	5	5	4.6	12	10	9	10.3
所要時間	1	1	1	1	1	1	1	1
バグの検出率	14.5	12.5	10.2	12.4	18.8	18.5	18.0	18.4
検出したバグの種類数	2	2	3	2.3	3	3	3	3
バグの検出効率	4	5	5	4.6	12	10	9	10.3

検証

- 検証結果
  - ツールを用いた探索的テストはツールを用いなかった探索的テストに比べてバグの検出効率(バグの総検出数/所要時間)は2.2倍に増加した

表1.　比較結果

探索的テスト手法	ツール未使用				ツール使用			
被験者	A	B	C	平均	D	E	F	平均
全テストケース生成数	27	40	49	38.6	64	54	50	56
バグの総検出数	4	5	5	4.6	12	10	9	10.3
所要時間	1	1	1	1	1	1	1	1
バグの検出率	14.5	12.5	10.2	12.4	18.8	18.5	18.0	18.4
検出したバグの種類数	2	2	3	2.3	3	3	3	3
バグの検出効率	4	5	5	4.6	12	10	9	10.3

2.2倍

考察

- 検証結果より、単位時間当たりのバグ検出数は2.2倍に増加したため
  テストエンジニアのセンスのブーストによって探索的テストの効率を
  向上させることができたと考えられる

- 探索的テストの重要なポイントの1つである学習は
  検証結果に大きな影響を与えるため、
  同じ被験者を2度扱うことができなかった
  - 被験者が本来持つ能力が検証結果に影響を与えている可能性がある
  - 被験者には本来持つ能力を一定にするためにテスト経験のない大学生を選んだ
    - 同じグループの被験者間でもテストケース生成数に大きなばらつきが発生しているため、
      被験者の能力や性格の差が検証結果に影響を与えている可能性がある

考察

- 本研究ではWord2vecのコーパスに日本語 wikiを用いたが、
  コーパスが変れば同じ単語を入力しても違った出力結果となる
  - 同じように探索的テストを行ってもコーパスが違うと再現性が保たれない
    ことに注意する必要がある
  - テスト 対象システムのドメインに合わせてコーパスを選択しないと、
    効果を示さない可能性がある

まとめ

- 本研究では、テストエンジニアのセンスをツールによって
  ブーストさせることでバグ検出数を増加させる手法を提案した

- テストエンジニアがツールによる補助で、
  普段より良い結果を得られるようにするというコンセプト

- 結果として、提案手法によってテストエンジニアは
  多くのバグを検出できる確率の高いテストケースを生成し、
  単位時間当たりのバグ検出数を増加させることができた