

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH**

KHOÁ LUẬN TỐT NGHIỆP

**XÂY DỰNG HỆ THỐNG TỰ ĐỘNG
TRẢ LỜI CÂU HỎI TRẮC NGHIỆM
TIẾNG ANH DẠNG ĐIỀN KHUYẾT**



Giảng viên hướng dẫn: **TS. NGUYỄN ANH TUẤN**

ThS. NGUYỄN VĂN TOÀN

Sinh viên thực hiện: **LÊ QUANG KHẢI - 09520134**

Lớp: **CNTN04**

TP. Hồ Chí Minh, tháng 7 năm 2013

MỞ ĐẦU

Tiếng Anh hiện nay là ngôn ngữ phổ biến nhất trên thế giới, do đó sử dụng tiếng Anh trôi chảy, lưu loát là một lợi thế rất lớn trong quá trình hội nhập, tiếp xúc với nguồn tri thức rộng lớn của thế giới. Xuất phát từ những lợi ích nhãn tiền cũng như các chính sách giáo dục của nhà nước, nhu cầu học tập, trau dồi kỹ năng sử dụng tiếng Anh trong giới học sinh, sinh viên hay thậm chí là nhân viên công sở đang ngày càng trở nên lớn hơn, nhiều hơn. Một trong các phương pháp căn bản trong quá trình học tiếng Anh đó là làm các bài tập về ngữ pháp. Để quá trình học tập đạt hiệu quả tốt hơn, người học nên có một người thầy, người bạn để hỗ trợ trong những lúc gặp khó khăn với các câu hỏi hóc búa. Tuy nhiên, không phải lúc nào cũng có người hỗ trợ cho chúng ta khi cần, đặc biệt là đối với những người có thói quen tự học, đây là một trong các động lực thúc đẩy chúng tôi thực hiện khóa luận tốt nghiệp với đề tài “Xây dựng hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết”.

Nội dung của đề tài này dựa trên các kiến thức đã được trang bị trong suốt 4 năm học đại học, đặc biệt là 2 năm cuối với những kiến thức có tính chuyên môn cao như máy học, xử lý ngôn ngữ tự nhiên, biểu diễn tri thức... Hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết áp dụng cách giải quyết của bài toán kiểm tra ngữ pháp thuộc lĩnh vực xử lý ngôn ngữ tự nhiên làm thành phần cốt lõi, sử dụng các kiến thức về máy học, biểu diễn tri thức và trí tuệ nhân tạo để hỗ trợ, nâng cao hiệu suất cho ứng dụng.

Bắt kịp xu thế thời đại của công nghệ di động, hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh được cài đặt và ưu tiên phát triển trên các thiết bị thông minh. Từ sự hỗ trợ tốt về mặt công nghệ, kiến trúc nhẹ nhàng và tính đơn giản trong khâu lập trình, chúng tôi chọn hệ điều hành Windows Phone của Microsoft làm nền tảng triển khai đầu tiên của ứng dụng. Hệ thống do chúng tôi xây dựng đã cho những kết quả khả quan bước đầu và hứa hẹn sẽ mang lại những lợi ích thiết thực cho người dùng.

Từ khóa: grammar checker, natural language processing, n-grams, windows phone, wcf, entity framework.

LỜI CẢM ƠN

Lời đầu tiên chúng em xin được bày tỏ lòng biết ơn sâu sắc nhất tới TS. Nguyễn Anh Tuấn, khoa Mạng máy tính và Truyền Thông và ThS. Nguyễn Văn Toàn, trưởng phòng công tác sinh viên, Đại học Công nghệ Thông tin, hai người thầy đã giúp đỡ em rất nhiều trong suốt thời gian thực hiện khóa luận tốt nghiệp.

Tiếp theo em xin được bày tỏ lòng biết ơn đến các giảng viên khoa Khoa học máy tính cũng như các giảng viên của các khoa khác trong trường Đại học Công nghệ Thông tin – ĐH Quốc Gia TP HCM. Các thầy cô đã chỉ bảo, định hướng, trang bị kiến thức chuyên môn và luôn tạo điều kiện học tập tốt nhất cho chúng em trong suốt những năm học đại học, đặc biệt là khoảng thời gian làm khóa luận.

Cùng với sự giúp đỡ của các giảng viên trong trường, sự hỗ trợ từ các bạn sinh viên lớp CNTN04 cũng thực sự có ý nghĩa với bản thân tôi trong suốt hai năm qua.

Lời cảm ơn cuối cùng xin được gửi đến cha, mẹ - những người đã luôn hỗ trợ, động viên tôi trong những thời điểm khó khăn, tạo điều kiện tốt nhất để tôi hoàn thành khóa luận này.

TP. Hồ Chí Minh, ngày 16 tháng 7 năm 2013

Lê Quang Khải

[illegible]

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Mục lục

Mục lục	v
Danh sách hình vẽ	viii
Danh sách bảng	x
1 GIỚI THIỆU	1
1.1 Tên đề tài	1
1.2 Nội dung và giới hạn đề tài	1
1.2.1 Nội dung đề tài	1
1.2.2 Giới hạn đề tài	2
1.3 Cấu trúc luận văn	2
2 TỔNG QUAN	4
2.1 Kiến thức nền tảng trong xử lý ngôn ngữ tự nhiên	4
2.1.1 Ngữ pháp của một ngôn ngữ	4
2.1.2 Đặc trưng ngôn ngữ	4
2.1.3 Token, tokenization, tokenizer	5
2.1.4 n-grams	5
2.1.5 Ngữ liệu	6
2.1.6 Gán nhãn ngữ nghĩa	7
2.1.7 Gán nhãn bằng phương pháp kết hợp	8
2.1.8 Tagset	9
2.1.9 Các chương trình PoS tagger	10
2.2 Bài toán kiểm tra ngữ pháp	11
2.2.1 Giới thiệu	11
2.2.2 Ứng dụng của bài toán kiểm tra ngữ pháp	12
2.3 Khảo sát, các công trình liên quan	12
2.3.1 Phương pháp hệ luật dẫn	12

MỤC LỤC

2.3.2	Phương pháp thống kê	13
2.4	Tổng kết chương	15
3	MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ	16
3.1	Ý tưởng chính	16
3.2	Thống kê n-grams kết hợp gom nhóm chủ ngữ	16
3.3	Thu thập dữ liệu	18
3.3.1	Giải thuật chọn đáp án	24
4	HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH	29
4.1	Xây dựng Web Service	29
4.1.1	Giới thiệu SOA (Service Oriented Application)	29
4.1.2	Tại sao phải sử dụng SOA	31
4.1.3	Giới thiệu Windows Communication Foundation	32
4.1.4	Kiến trúc của WCF	33
4.1.5	Xây dựng AnswerService	34
4.1.6	Gán nhãn từ loại bằng SharpNLP	39
4.1.7	Chạy thử và kiểm tra	41
4.2	Sử dụng Hadoop để lấy dữ liệu từ Google Books n-grams	44
4.2.1	Giới thiệu Hadoop	44
4.2.2	Giới thiệu HDFS	44
4.2.3	Xử lý song song với MapReduce	45
4.2.4	Xây dựng Hadoop Cluster	47
4.2.5	Thu thập dữ liệu từ Google Books n-grams bằng Hadoop	48
4.3	Ứng dụng giải bài tập trắc nghiệm tiếng Anh trên Windows Phone	52
4.3.1	Giới thiệu hệ điều hành Windows Phone	52
4.3.2	Kiến trúc Windows Phone	54
4.3.3	Mô hình ba lớp Model-View-View Model (MVVM)	56
4.3.4	Xây dựng ứng dụng	58
4.3.4.1	Model	58
4.3.4.2	View Model	60
4.3.4.3	View	61
4.3.4.4	Screenshots	62
4.3.5	Mở rộng hệ thống trên các nền tảng khác	65
4.3.5.1	Ứng dụng trên iPhone	65

MỤC LỤC

5	KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	67
5.1	Dữ liệu khảo sát	67
5.2	Chương trình khảo sát tự động	68
5.3	Thực hiện khảo sát	69
5.3.1	Khảo sát 1 – Độ chính xác của các phương án chọn n-grams	69
5.3.2	Khảo sát 2 - Sự phụ thuộc về độ lớn cơ sở dữ liệu với độ chính xác	71
5.3.2.1	Khảo sát đợt một	71
5.3.2.2	Khảo sát đợt hai	71
5.3.3	Khảo sát độ chính xác của từng loại câu hỏi	72
5.4	Đánh giá	72
5.5	Kết luận	73
5.6	Hướng phát triển	74
5.6.1	Bổ sung n-grams cho cơ sở dữ liệu	74
5.6.2	Ứng dụng hệ luật dẫn để cải thiện độ chính xác	75
5.6.3	Cải thiện ứng dụng người dùng, phát triển ứng dụng trên các nền tảng khác	75
	Tài liệu tham khảo	76

Danh sách hình vẽ

2.1	Hệ thống NLP của Microsoft	13
3.1	Cấu trúc một file –hepper.xml	19
3.2	Cấu trúc một file –s.xml	20
3.3	Hàm GetSentences	21
3.4	Hàm ProcessSentences	22
3.5	Hàm GetGrams	23
3.6	Cấu trúc bảng lưu trữ các bigrams và trigrams	23
3.7	Bảng Total	24
3.8	Giao diện chương trình xử lý và thu thập dữ liệu từ OANC	24
3.9	Mô phỏng quá trình đưa ra gợi ý đáp án cho câu hỏi	25
3.10	Sơ đồ khối quy trình lựa chọn phương án cho câu hỏi nhập vào	26
3.11	Phương án chỉ lấy các bigrams	27
3.12	Phương án chỉ lấy các trigrams	27
3.13	Phương án kết hợp sử dụng bigrams và trigrams	28
4.1	Mô hình kiến trúc SOA	30
4.2	Mô hình thống nhất thay thế cho các công nghệ phân tán trước đó	32
4.3	Kiến trúc của WCF	34
4.4	UML Class trong WCF Service	35
4.5	Truy vấn cơ sở dữ liệu để lấy tổng số lần xuất hiện của n-grams	38
4.6	Code map thể hiện các phương thức được gọi để trả lời câu hỏi	38
4.7	Code map các phương thức được gọi để học từ gợi ý người dùng	39
4.8	Sắp xếp các file, folder cho dữ liệu huấn luyện của SharpNLP	40
4.9	Quy trình gán nhãn từ loại	40
4.10	Lớp SharpNLP	41
4.11	Giao diện công cụ WCF Test Client	42
4.12	Test AnswerService với công cụ WCF Test Client	43
4.13	Kiến trúc HDFS	45

DANH SÁCH HÌNH VẼ

4.14	Quy trình thực thi MapReduce	46
4.15	Kiến trúc Hadoop và các thành phần liên quan	47
4.16	Hadoop cluster được xây dựng trong mạng nội bộ	48
4.17	Quản lý tiến độ thực hiện MapReduce trên giao diện console	49
4.18	Quản lý tiến độ thực hiện MapReduce trên giao diện Web	50
4.19	Thông kê dữ liệu của MapReduce job đang chạy	51
4.20	Thông kê tiến độ thực hiện bằng đồ thị	52
4.21	Lịch sử phát triển các hệ điều hành di động của Microsoft	53
4.22	Giao diện Home của HDH Windows Phone	54
4.23	Kiến trúc hệ điều hành Windows Phone 7	55
4.24	Kiến trúc hệ điều hành Windows Phone 8	56
4.25	Kiến trúc MVVM	57
4.26	Quy trình hoạt động của mô hình MVVM	57
4.27	Quy trình xử lý của hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh	58
4.28	Lược đồ UML các lớp trong Model	59
4.29	Gọi AnswerService	59
4.30	Các class trong View Model layer	60
4.31	Các class trong View layer	61
4.32	ChoiceControl	61
4.33	A. Giao diện portrait. B. Giao diện landscape	62
4.34	A. Sau khi nhập xong câu hỏi và các phương án trả lời. B. Ứng dụng đang chờ câu trả lời từ AnswerService	63
4.35	A. đưa ra đáp án đề xuất cho câu hỏi. B. Người dùng chọn lại đáp án nếu đáp án đề xuất không chính xác	64
4.36	Màn hình credit	64
4.37	A. Màn hình nhập các phương án lựa chọn. B. Màn hình chờ trong lúc tìm đáp án	65
4.38	A. Màn hình hiển thị đáp án được chọn. B. Màn hình cho phép người dùng đề xuất đáp án khác	66
5.1	Cấu trúc XML để lưu trữ các bộ đề thi	68
5.2	Chương trình khảo sát tự động	69

Danh sách bảng

2.1	Ví dụ n-grams	6
2.2	Ví dụ các n-grams nhân từ loại	14
2.3	Các loại cụm từ	14
3.1	Phân loại các nhóm chủ ngữ	17
3.2	Ví dụ phân loại nhóm chủ ngữ cho câu	17
3.3	Thống kê dữ liệu trong OANC	18
3.4	Cấu trúc tổ chức file của một chủ đề trong OANC	19
3.5	Ví dụ câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết	25
4.1	Các phương thức sử dụng trong AnswerService	37
4.2	Biểu diễn dữ liệu trong Google Book n-grams file	48
4.3	Mapper và Reducer để thu thập n-grams	49
5.1	Thống kê số lượng câu hỏi các bộ đề thi	67
5.2	Kết quả khảo sát trên đề ôn thi bằng A tiếng Anh	69
5.3	Kết quả khảo sát trên đề ôn thi bằng B tiếng Anh	70
5.4	Kết quả khảo sát trên đề ôn thi bằng C tiếng Anh	70
5.5	Kết quả khảo sát trên đề ôn thi TOELF	70
5.6	Kết quả khảo sát trên đề thi đại học khối D từ năm 2009 đến năm 2012	71
5.7	Kết quả khảo sát lần 1 với cơ sở dữ liệu nhỏ	71
5.8	Kết quả khảo sát lần 2 trên cơ sở dữ liệu lớn hơn	72
5.9	Kết quả khảo sát các loại câu hỏi trên 100 câu hỏi từ đề thi đại học	72
5.10	Kết quả khảo sát các loại câu hỏi trên 200 câu hỏi đầu tiên trong đề ôn thi bằng C	73

Chương 1

GIỚI THIỆU

1.1 Tên đề tài

Xây dựng hệ thống tự động trả lời câu hỏi trắc nghiệm tiếng Anh điền khuyết.

1.2 Nội dung và giới hạn đề tài

1.2.1 Nội dung đề tài

Thường xuyên tự học và tự rèn luyện bằng các bài trắc nghiệm tiếng Anh là một trong những phương pháp cơ bản để tự nâng cao trình độ tiếng Anh cũng như bổ sung kiến thức trước các kỳ thi lấy chứng chỉ quan trọng như thi lấy bằng A, B, C, các chứng chỉ quốc tế như TOEIC, TOELF... Tuy nhiên, điều khó khăn cho những người tự học là việc kiểm tra tính chính xác của đáp án của các câu trả lời của mình. Bên cạnh đó, hiện nay đã có một số công cụ có khả năng hỗ trợ giải các loại bài tập tự động do người Việt xây dựng như các ứng dụng giải bài tập Toán, Lý (các hệ thống áp dụng mạng tính toán COKB của thầy Đỗ Văn Nhơn), hệ thống giải bài tập lập phương trình Hóa học của trang tìm kiếm coccoc.com. Thế nhưng vẫn chưa có hệ thống nào thực hiện việc giải bài tập trắc nghiệm cho môn tiếng Anh - một môn học rất quan trọng đối với các bạn học sinh cũng như sinh viên. Xuất phát từ hai vấn đề trên, cùng với việc nghiên cứu bài toán kiểm tra ngữ pháp trong lĩnh vực xử lý ngôn ngữ tự nhiên, chúng tôi đề xuất ý tưởng xây dựng một hệ thống có thể tự động giải các bài tập trắc nghiệm tiếng Anh.

Việc ứng dụng bài toán kiểm tra ngữ pháp để giải các dạng bài tập trắc nghiệm về ngôn ngữ thực tế đã được nhắc đến trong một vài công trình nghiên cứu trước đây. Tuy nhiên cho đến thời điểm hiện tại vẫn chưa có công trình nào công bố ứng dụng thực tiễn cho phép người dùng sử dụng đại trà. Bên cạnh đó, trong quá trình nghiên cứu đề tài, chúng tôi cũng đã phát

Chương 1. GIỚI THIỆU

hiện và bổ sung một vài đặc điểm cải tiến nhỏ trong việc kiểm tra ngữ pháp.

Để hiện thực hóa ý tưởng, chúng tôi chọn xây dựng một hệ thống xử lý tập trung theo xu hướng rất thịnh hành hiện nay là Service Oriented Application (SOA). Cùng với đó, điện thoại thông minh đang có sự phát triển rất mạnh mẽ, giá tiền của một chiếc điện thoại đang rẻ đi từng ngày, giúp cho số lượng người được tiếp cận với công nghệ mới, với internet ngày càng nhiều. Chính vì vậy chúng tôi quyết định xây dựng ứng dụng trên nền tảng di động mà phiên bản đầu tiên là trên hệ điều hành Windows Phone của Microsoft và trong tương lai sẽ mở rộng ra các nền tảng phổ biến khác như iOS hay Android.

Tóm lại, khóa luận này sẽ thực hiện các công việc sau:

1. Cài đặt phương pháp gom nhóm chủ ngữ đối với ngữ pháp tiếng Anh.
2. Thống kê n-grams từ bộ ngữ liệu Open American National Corpus.
3. Xây dựng thuật giải kiểm tra ngữ pháp tiếng Anh dựa trên phương pháp thống kê.
4. Áp dụng bài toán kiểm tra ngữ pháp tiếng Anh để giải câu hỏi trắc nghiệm điền khuyết tiếng Anh.
5. Xây dựng Web Service và ứng dụng di động để hiện thực ý tưởng về hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết.

1.2.2 Giới hạn đề tài

Bài tập trắc nghiệm tiếng Anh có nhiều dạng khác nhau, ví dụ như:

- Bài tập điền khuyết
- Tìm lỗi sai trong câu
- Đọc hiểu văn bản và chọn câu đúng nhất
- Chọn từ thích hợp cho đoạn văn
- Chọn từ có trọng âm khác với các từ còn lại

Hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh được trình bày trong khóa luận sẽ chỉ giải quyết bài tập trắc nghiệm điền khuyết. Nghĩa là cho một câu tiếng Anh với một vị trí bị khuyết cùng với các phương án để thế vào vị trí chỗ khuyết đó, hệ thống của chúng tôi sẽ xử lý và đề xuất một phương án được cho là đúng. Các dạng bài tập còn lại hay bài tập điền khuyết có nhiều hơn một chỗ trống không thuộc phạm vi nghiên cứu của luận văn này.

1.3 Cấu trúc luận văn

Luận văn này được trình bày với cấu trúc như sau:

Chương 1. GIỚI THIỆU

- **Chương 2:** khái quát các kiến thức liên quan đến lĩnh vực xử lý ngôn ngữ tự nhiên và bài toán kiểm tra ngữ pháp của một ngôn ngữ; khảo sát các công trình thực hiện việc kiểm tra ngữ pháp đã được công bố, rút ra những thiếu sót của các hướng tiếp cận.
- **Chương 3:** đề xuất mô hình tiếp cận cho bài toán kiểm tra ngữ pháp bằng thống kê; trình bày phương pháp gom nhóm chủ ngữ trong ngữ pháp tiếng Anh để hỗ trợ cho việc thống kê n-grams
- **Chương 4:** hiện thực hóa ý tưởng, áp dụng các công nghệ có sẵn để xây dựng hệ thống theo hướng đã đề ra.
- **Chương 5:** khảo sát kết quả của việc nghiên cứu và cài đặt hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết; rút ra các đánh giá, nhận xét và kết luận, đồng thời chỉ ra các hướng phát triển có thể thực hiện trong tương lai.

Chương 2

TỔNG QUAN

2.1 Kiến thức nền tảng trong xử lý ngôn ngữ tự nhiên

2.1.1 Ngữ pháp của một ngôn ngữ

Con người giao tiếp với nhau thông qua ngôn ngữ tự nhiên dưới dạng văn viết và lời nói. Cú pháp của một ngôn ngữ mô tả cách kết hợp các từ ngữ riêng biệt để tạo thành câu có nghĩa. Hình thái của một ngôn ngữ chính là sự biến đổi của các từ tùy thuộc vào các yếu tố khác như thời gian, số lượng, giới tính của từ. Như vậy, ta có thể định nghĩa ngữ pháp của một ngôn ngữ tự nhiên là một tập hợp các luật về cú pháp và sự biến đổi của các từ (hình thái từ) trong ngôn ngữ để tạo thành một câu có nghĩa.

Các ngôn ngữ khác nhau đều có cấu trúc ngữ pháp khác nhau. Cụ thể, nếu hai ngôn ngữ có cú pháp hoặc hình thái từ khác nhau thì chúng là hai ngôn ngữ riêng biệt. Ví dụ, tiếng Anh và tiếng Mỹ là hai ngôn ngữ rất giống nhau, hầu hết mọi người đều không phân biệt nó là hai ngôn ngữ khác nhau. Tuy nhiên, chỉ cần có một điểm khác nhau về ngữ pháp, chúng vẫn được xem như là hai ngôn ngữ khác nhau theo định nghĩa về ngữ pháp đã nêu ở trên.

2.1.2 Đặc trưng ngôn ngữ

Mỗi ngôn ngữ khác nhau về đặc điểm hình thái. Độ phức tạp của một ngôn ngữ thường phụ thuộc vào sự biến đổi của các tính chất như theo giới tính từ, theo số lượng, theo ngữ cảnh. Ví dụ tiếng Iceland và tiếng Đức phức tạp hơn nhiều so với tiếng Anh, trong mọi trường hợp đều có thể có nhiều lựa chọn khác nhau để tạo thành câu.

Các ngôn ngữ tự nhiên khác nhau cũng có chiều dài từ và câu khác nhau. Ví dụ, đối với tiếng Đức, câu dài thường được sử dụng phổ biến nhưng trong tiếng Anh lại không được ưa thích. Chiều dài trung bình của từ cũng trong các ngôn ngữ cũng khác nhau.

Chương 2. TỔNG QUAN

2.1.3 Token, tokenization, tokenizer

Mỗi thành phần trong câu, bao gồm từ, số, dấu câu, từ viết tắt, ... đều được gọi là một token. Việc tách câu thành các token gọi là tokenization và được thực hiện bởi tokenizer. Tokenization là bước quan trọng trong quá trình gán nhãn từ loại.

Có nhiều cách để tokenize một đoạn văn bản, các kiểu viết tắt như **can't** có thể được xem như một token, nhưng cũng có thể được tách thành hai từ **can not**. Một trường hợp tương tự khác đó là tokenize một chuỗi số. Cách biểu diễn một số lớn thường có nhiều cách khác nhau, ví dụ số 112500 có thể được viết liên tục, không có dấu xen giữa. Tuy nhiên trong một số ngôn ngữ, người ta thường thêm vào dấu phẩy hoặc chấm để giúp người đọc dễ phân biệt hơn, ví dụ như 1.125.000 hay 1,125,000.

Khó khăn lớn nhất trong quá trình tokenization là phân tách câu và từ. Một vài ký tự biểu tượng như “ . ? ! ” có thể xem như mốc đánh dấu kết thúc một câu trong các ngữ sử dụng mẫu tự Latin hoặc Cyrillic alphabet. Ta có thể thấy một ký tự biểu tượng có thể đóng nhiều vai trò khác nhau trong ngôn ngữ. Ví dụ như trong tiếng Anh, dấu chấm có thể được dùng để kết thúc một câu, dùng để tách đơn vị trong chữ số hay dùng trong từ viết tắt. Điều này làm cho việc phân tách một câu trở nên khó khăn hơn. Ta có thể giải quyết vấn đề trên bằng cách sử dụng một danh sách các từ viết tắt để phân biệt với trường hợp dấu chấm kết thúc câu. Tuy nhiên cách này vẫn chưa giải quyết được trường hợp nhập nhàn khi một câu có từ viết tắt ở cuối câu.

Một khó khăn khác cần giải quyết đó là chia tách từ. Thông thường các từ sẽ được phân biệt với nhau bằng khoảng trắng. Tuy nhiên cũng có một vài trường hợp ngoại lệ, ví dụ như gán nhãn cho một cụm từ. Giả sử ta có cụm từ “to kick the bucket”, nếu xem khoảng trắng là ký tự phân biệt thì ta sẽ có 4 nhãn khác nhau. Tuy nhiên, nghĩa của cả cụm từ trên là “to die” nên cho chính xác thì ta sẽ chỉ sử dụng một nhãn cho cả cụm từ, trong trường hợp này là động từ thay vì bốn nhãn từ. Trường hợp các ngôn ngữ dùng ký tự tượng hình như tiếng Trung hoặc tiếng Nhật. Hai ngôn ngữ này vẫn dùng dấu chấm để phân biệt câu, nhưng các từ lại không nhất thiết phải được phân biệt nhờ vào khoảng trắng. Có thể thấy tokenization không phải là một công việc đơn giản và kết quả của quá trình này ảnh hưởng trực tiếp đến kết quả của các công đoạn sau, ví dụ như gán nhãn ngữ nghĩa.

2.1.4 n-grams

Trong lĩnh vực ngôn ngữ học máy tính và xác suất thống kê, n-gram là một chuỗi gồm n token liên tiếp được tách ra từ chuỗi lớn hơn [1, 5]. Ta không phân biệt loại token đó là chữ, là số, là từ viết tắt hay dấu câu. Thông thường, người ta thu thập n-gram từ một đoạn văn bản hoặc từ các bộ ngữ liệu.

Chương 2. TỔNG QUAN

1-gram	2-gram	3-gram
To, be, or, not, to, be	To be, be or, or not, not to, to be	To be or, be or not, or not to, not to be

Bảng 2.1: Ví dụ n-grams

Kích thước của n-gram thường nằm trong khoảng từ 1 đến 5. Ứng với mỗi giá trị n sẽ có tên gọi khác nhau, ví dụ:

- $n = 1$ gọi là unigram
- $n = 2$ gọi là bigrams
- $n = 3$ gọi là trigrams
- $n = 4$ gọi là tetragrams
- $n = 5$ gọi là pentagrams
- Với n lớn hơn, ta gọi tên của các gram này bằng cách đọc kết hợp giá trị của n với từ “gram”.

Ví dụ: cho một chuỗi có nội dung như sau: “... to be or not to be”, với đơn vị nhỏ nhất là từ, ta thu thập được các n-gram như ở bảng 2.1.

Một câu sẽ có $k(n-1)$ n-grams với k là chiều dài của câu tính luôn cả dấu câu và n là số token trong một n-grams, hay nói cách khác chính là giá trị n .

Ngoài ra, một biến thể khác của n-grams đó là thay vì bao gồm n token liên tiếp trong câu, n-grams dạng này bao gồm nhãn ngữ nghĩa của các tokens. Để phân biệt giữa hai loại n-grams này, ta chia ra thành n-grams of tokens và n-grams of tags. Ví dụ, ta có “modern houses are” là trigrams of tokens và “adjective noun verb” là trigrams of tags.

2.1.5 Ngữ liệu

Ngữ liệu (corpus) là tập hợp các văn nói và văn viết của một ngôn ngữ tự nhiên nhất định. Dữ liệu của các ngữ liệu này thường được số hóa để lưu trữ trên máy tính và máy có thể đọc hiểu được [5]. Ngữ liệu bao gồm các thành phần sau:

- Dữ liệu văn bản
- Các siêu dữ liệu (meta data) mô tả bổ sung cho dữ liệu văn bản
- Các chú thích về từ vựng liên quan đến dữ liệu văn bản.

Mỗi ngôn ngữ khác nhau đều có một hoặc nhiều nhiều bộ ngữ liệu, trong đó số lượng nhiều nhất là ngữ liệu tiếng Anh, ngoài ra còn có hai loại ngữ liệu đặc biệt khác là Google n-grams và các ngữ liệu về lỗi ngữ pháp. Các bộ ngữ liệu phổ biến thường được sử dụng hiện

Chương 2. TỔNG QUAN

này bao gồm:

American National Corpus (ANC) – hiện có hơn 20 triệu từ vựng tiếng Anh mà người Mỹ sử dụng và được quản lý bởi Linguistic Data Consortium. Dự án này vẫn còn đang trong quá trình phát triển và dự kiến lúc kết thúc sẽ có hơn 100 triệu từ.

British National Corpus (BNC) – tương tự như bộ ngữ liệu ANC nhưng là tiếng Anh của người Anh. Phiên bản hoàn thiện mới nhất năm 2007 chứa khoảng 100 triệu từ. Theo trang chủ của BNC thì bộ ngữ liệu này được xây dựng từ nhiều nguồn, kể cả văn viết và các đoạn thu âm lời nói tính từ thế kỷ 20.

Brown Corpus – một sự lựa chọn khác thay cho ANC và BNC là Standard Corpus do Present-Day American English xây dựng (còn gọi là Brown Corpus). Bộ ngữ liệu này chứa khoảng 1 triệu từ được thu thập từ các ấn phẩm bằng tiếng Anh của Mỹ trong suốt năm 1961. Tính đến thời điểm hiện tại Brown Corpus đã có 6 phiên bản, tất cả các phiên bản này đều giống nhau về nội dung nhưng khác nhau về định dạng và cách tổ chức.

Wortschatz Universität Leipzig – đây là bộ ngữ liệu do trường đại học Leipzig xây dựng và hỗ trợ đến 18 ngôn ngữ khác nhau như tiếng Anh, tiếng Đức, Icelandic. Nguồn dữ liệu của bộ ngữ liệu này được thu thập từ tạp chí cũng như lấy ngẫu nhiên từ các văn bản trên Internet. Ngữ liệu tiếng Đức có kích thước khoảng 30 triệu câu, ngữ liệu tiếng Anh có 10 triệu câu và Icelandic có 1 triệu câu.

NEGRA corpus – một bộ ngữ liệu tiếng Đức thông dụng là NEGRA corpus, bao gồm 355,096 tokens (khoảng 20 ngàn câu) được thu thập từ các tạp chí tiếng Đức. Bộ ngữ liệu này cũng đã được gán nhãn ngữ liệu và chú thích cấu trúc cú pháp.

Google n-grams – bộ ngữ liệu này khác biệt so với các bộ ngữ liệu còn lại ở chỗ nó chỉ chứa danh sách các n-grams chứ không phải văn bản. Nguồn dữ liệu của Google n-grams được thu thập từ các trang web tiếng Anh, và được phân tích thành unigrams, bigram, ... đến pentagrams. Mỗi n-grams đều có thông tin về tần số xuất hiện.

Error corpus – đây là bộ ngữ liệu gồm các câu viết cố ý sai về ngữ pháp và câu đúng tương ứng.

2.1.6 Gán nhãn ngữ nghĩa

Vai trò chính của việc gán nhãn ngữ nghĩa (part-of speech (PoS) tagging) là phân lớp chính xác và xác định hình thái của các token trong văn bản. Kết quả của quá trình này là một văn bản với các token đã được chú thích thêm về loại từ cũng như hình thái của nó. Đây là bước tiền xử lý quan trọng trước khi thực hiện tiếp các bước tiếp theo trong các ứng dụng xử lý ngôn ngữ tự nhiên, ví dụ như trong kiểm tra ngữ pháp, trích xuất thông tin, dịch máy. Tuy nhiên, không phải tất cả các từ đều được gán nhãn chính xác, trong nhiều trường hợp có thể xảy ra các trường hợp nhập nhằng do một từ có thể đóng nhiều vai trò trong câu. Do đó, hiện

Chương 2. TỔNG QUAN

nay vẫn còn đang có nhiều vấn đề cần được giải quyết để tìm ra giải pháp tốt nhất trong việc gán nhãn ngữ nghĩa này.

Các nhãn ngữ nghĩa là các chuỗi ký tự khác nhau, tập hợp các nhãn ngữ nghĩa này gọi là *tagset*. Hiện nay có một vài bộ *tagset* đang được sử dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên. Quá trình gán nhãn ngữ nghĩa được thực hiện bởi *tagger*, tính năng chính của nó là loại bỏ sự nhập nhằng về hình thái của một từ để đưa ra nhãn ngữ nghĩa chính xác của từ đó. Hiện nay có nhiều phương pháp được sử dụng để loại bỏ sự nhập nhằng này.

Thuật ngữ *tagging accuracy* được sử dụng để mô tả tỉ lệ chính xác của quá trình gán nhãn ngữ nghĩa, được tính bằng số lượng nhãn được gán chính xác cho các token đem chia cho tổng số token có trong văn bản. Thông thường, tất cả các ngôn ngữ khác nhau đều có *tagger* để gán nhãn ngữ nghĩa. Các ngôn ngữ khác nhau cũng có số lượng *tagger* hỗ trợ không giống nhau, ví dụ rõ ràng tiếng Anh sẽ có nhiều *tagger* hơn so với tiếng Việt do số lượng người nói tiếng Anh rất nhiều, ngoài ra độ phức tạp của các hình thái trong ngôn ngữ cũng ảnh hưởng đến số lượng *tagger* hỗ trợ cho ngôn ngữ. Độ chính xác của *tagger* có hình thái phức tạp cũng thấp hơn so với tiếng Anh.

2.1.7 Gán nhãn bằng phương pháp kết hợp

Gán nhãn bằng phương pháp kết hợp là phương pháp sử dụng đồng thời nhiều bộ *tagger* trên một văn bản, kết quả của mỗi chương trình gán nhãn sẽ được tổng hợp và xem xét theo tuy chí cụ thể để đưa ra nhãn ngữ nghĩa chính xác nhất cho mỗi token. Thông thường kết quả của phương pháp gán nhãn kết hợp sẽ có độ chính xác cao hơn so với khi sử dụng đơn lẻ một chương trình nhất định. Nguyên nhân chủ yếu là do các chương trình gán nhãn khác nhau thường đưa ra các lỗi khác nhau và ta có thể khai thác sự khác biệt đó để lấy được kết quả tốt hơn. Khi xây dựng chương trình gán nhãn kết hợp, điều quan trọng cần ghi nhớ là các *tagger* được sử dụng phải sử dụng các phương thức gán nhãn khác nhau [5].

Có nhiều chiến lược kết hợp các *tagger* khác nhau, các chiến lược này được gọi là giải thuật kết hợp:

Simple voting – đây là giải thuật kết hợp đơn giản nhất. Tất cả *tagger* riêng biệt được xếp hạng bình đẳng về kết quả gán nhãn cho một token nhất định. Lấy tổng giá trị đánh giá của từng *tagger*, nhãn có tổng cao nhất sẽ được chọn làm kết quả cuối cùng cho token đó. Trong trường hợp hai tag có giá trị tổng bằng nhau, ta sẽ chọn nhãn ngữ nghĩa được đưa ra bởi bộ *tagger* có độ chính xác trung bình cao hơn.

Weighted voting – tương tự như *simple voting* nhưng cho phép mỗi *tagger* đưa ra các trọng số khác nhau. Ví dụ với *tagger* thường cho độ chính xác cao hơn sẽ có trọng số cao hơn khi bình chọn. Giá trị của trọng số cũng được tính vào giá trị tổng và nhãn nào có giá trị tổng hợp cao nhất sẽ được chọn. Trong trường hợp kết quả bình chọn bằng nhau (thường hiếm khi

Chương 2. TỔNG QUAN

xảy ra khi sử dụng giải thuật này), ta cũng sử dụng giải pháp tương tự như simple voting.

2.1.8 Tagset

Một bộ tagset là tập hợp các nhãn ngữ nghĩa khác nhau. Hiện nay có nhiều bộ tagset khác nhau và được lựa chọn sử dụng tùy vào ngôn ngữ. Các bộ tagset thông dụng hiện nay bao gồm:

Penn Treebank tagset – đây là một trong những bộ tagset quan trọng nhất trong tiếng Anh, được xây dựng bởi Penn Treebank Project. Tagset này chứa 36 nhãn từ loại và 12 nhãn cho các dấu câu cũng như ký tự tiền tệ. Penn Treebank Project thuộc đại học Pennsylvania, tất cả các dữ liệu của Treebank được công bố thông qua tổ chức Linguistic Data Consortium.

Stuttgart-Tübingen Tagset – đây là tagset thông dụng được dùng cho tiếng Đức. Tagset này có tổng cộng 54 nhãn và được cấu trúc theo dạng phân cấp, trong đó 48 nhãn là nhãn từ loại, 6 nhãn còn lại biểu diễn các thành phần bổ sung như dấu câu hay các từ nước ngoài. STTS là kết quả tổng hợp của hai bộ tagset được phát triển bởi viện xử lý ngôn ngữ tự nhiên thuộc đại học Stuttgart và khoa ngôn ngữ học thuộc đại học Tübingen.

Icelandic Frequency Dictionary corpus tagset – Penn Treebank tagset hay STTS chỉ chứa số lượng nhãn ngữ nghĩa ít hơn nhiều so với các loại ngôn ngữ giàu hình thái, ví dụ bộ tagset chính của tiếng Iceland có đến khoảng 700 nhãn. Tagset này được xây dựng dựa trên bộ ngữ liệu Icelandic Frequency Dictionary. Các nhãn ngữ nghĩa không chỉ đơn thuần là nhãn từ loại biểu diễn một hình thái, thay vào đó mỗi ký tự trong nhãn sẽ biểu diễn một chức năng khác nhau. Ví dụ, ký tự đầu tiên được dùng để biểu diễn lớp của từ. Tùy vào lớp của từ đã được phân loại sẽ có một số lượng ký tự theo sau nhất định để chú thích các hình thái từ. Đối với danh từ có thể là giới tính của từ, số lượng... hay với động từ có thể là trọng âm, thì của từ...

Brown tagset và Münsteraner tagset – như đã nói ở trên, một bộ tagset thường chỉ được sử dụng trong một ngôn ngữ, tuy nhiên điều này không có nghĩa là một ngôn ngữ chỉ được phép có 1 tagset. Với tiếng Anh và tiếng Đức, có ít nhất 2 bộ tagset. Bên cạnh Penn Tree bank và STTS tagset có thể xem là các tagset chuẩn còn có Brown tagset và Münsteraner tagset. Hai tagset này đều định nghĩa nhiều nhãn ngữ nghĩa hơn so với Penn Treebank và STTS. Münsteraner tagset được xây dựng khá tương tự với bộ tagset của tiếng Iceland ở chỗ mỗi ký tự biểu diễn một đặc tính của từ.

Một vài bộ tagset còn định nghĩa cách tokenized văn bản trước khi đưa xử lý bởi tagger, tiêu biểu là Penn Treebank tagset.

Chương 2. TỔNG QUAN

2.1.9 Các chương trình PoS tagger

Các chương trình tagger khác nhau có thể cài đặt cơ chế gán nhãn khác nhau. Các phương pháp đã được áp dụng có thể kể đến như phương pháp dựa trên luật dẫn, mô hình Markov ẩn, phương pháp thống kê bằng cây quyết định, học dựa trên lỗi (error-driven transformation-based learning) hoặc mô hình maximum entropy. Các bộ tagger sử dụng các phương pháp gán nhãn kể trên:

Brill tagger – bộ gán nhãn được giới thiệu vào năm 1994 bởi Brill. Chương trình này sử dụng phương pháp máy học và cho kết quả khá cao. Quá trình training diễn ra như sau: hệ thống tự động gán một nhãn vào mỗi từ và sử dụng các luật đã được khai báo sẵn, các luật này sẽ được thay đổi thường xuyên theo chu kỳ. Nếu gặp phải một từ đã biết, nhãn có tần suất xuất hiện cao nhất sẽ được gán cho từ đó. Nếu là từ mới, hệ thống sẽ gán nhãn noun (danh từ) cho từ đó. Quá trình này được lặp lại nhiều lần để sửa lỗi cho đến khi ra được kết quả cuối cùng.

TnT – đây là bộ gán nhãn ứng dụng mô hình Markov ẩn dựa trên việc cài đặt giải thuật Viberti, là viết tắt của Trigrams'n'Tags. Hệ thống gán nhãn này không phụ thuộc vào một ngôn ngữ bất kỳ nào, các ngôn ngữ mới sẽ được huấn luyện trước khi thực hiện gán nhãn.

SVMTool - gán nhãn dựa vào Support Vector Machine, bộ gán nhãn này làm tốt hơn TnT Tagger, cho ra kết quả cao hơn ở cùng điều kiện. SVM đánh nhãn tiếng Anh trên ngữ liệu Wall Street Journal với độ chính xác là 97.2%

Stanford Tagger – hoạt động dựa trên mô hình entropy cực đại (maximum entropy). Tagger này sẽ học từ các văn bản đã được gán nhãn hoàn chỉnh, tính ra xác suất có điều kiện và gán các giá trị xác suất cho từng nhãn, giá trị này sẽ được sử dụng để đánh giá các kết quả. Stanford Tagger cũng là không phụ thuộc vào loại ngôn ngữ nào, được phát triển tại trường đại học Stanford.

TreeTagger – đây là bộ gán nhãn sử dụng phương pháp thống kê. Công cụ này được phát triển trong dự án TC tại viện nghiên cứu ngôn ngữ học thuộc đại học Stuttgart. TreeTagger đã được ứng dụng thành công trong việc gán nhãn cho nhiều ngôn ngữ như tiếng Đức và tiếng Anh. Hệ thống này cũng có thể dễ dàng áp dụng trên các ngôn ngữ khác nếu có bộ ngữ nghĩa huấn luyện phù hợp.

IceTagger – đây là công cụ gán nhãn dựa trên hệ luật dẫn. Các hệ thống gán nhãn dạng này thường chủ yếu dựa trên đặc trưng tự nhiên của ngôn ngữ, do đó không thể áp dụng trên nhiều ngôn ngữ như các phương pháp khác. IceTagger là hệ thống gán nhãn được sử dụng cho tiếng Iceland.

2.2 Bài toán kiểm tra ngữ pháp

2.2.1 Giới thiệu

Như đã định nghĩa ở trên, ngữ pháp của một ngôn ngữ tự nhiên được biểu diễn bằng các cú pháp và hình thái từ. Do đó, kiểm tra ngữ pháp có thể hiểu là việc kiểm tra tính chính xác của cú pháp và hình thái đối với ngôn ngữ đang xét. Công cụ được dùng để thực hiện việc kiểm tra ngữ pháp thường được gọi là *grammar checker* [11].

Có nhiều phương pháp khác nhau để kiểm tra tính chính xác về ngữ pháp trên một đoạn văn bản, bao gồm:

- **Pattern matching:** một trong những cách cơ bản nhất là so khớp mẫu. Phương pháp này hoạt động bằng cách lưu trữ song song các lỗi ngữ pháp thông dụng cùng với hình thái chính xác của lỗi ngữ pháp đó. Một câu hoặc một phần của câu sẽ được kiểm tra bằng cách so khớp với một số lỗi được lưu trong cơ sở dữ liệu. Nếu trùng khớp, lỗi đó sẽ được phát hiện và được sửa nhờ vào dữ liệu có sẵn. Phương pháp này cho độ hiệu quả cao khi gặp các mẫu đã được lưu trong cơ sở dữ liệu, tuy nhiên lại không có tính tổng quát. Mỗi lỗi mới, dù nhỏ đều phải được thêm vào cơ sở dữ liệu, ví dụ “He sell” và “He tell” khác nhau về loại lỗi nhưng phải đưa vào hai mẫu riêng biệt [6, 8].
- **Rule-based approach:** một phương pháp phổ biến hơn thường được dùng để kiểm tra ngữ pháp là dựa vào các luật. Một luật đơn giản có thể được dùng để phát hiện ra các lỗi dễ tìm thấy, tuy nhiên với câu càng phức tạp, luật cần dùng để phát hiện lỗi cũng có độ phức tạp cao hơn. Quay lại ví dụ về trường hợp “He tell” và “He sell” lúc này, ta có thể nhanh chóng định nghĩa một luật với nội dung đại khái như “động từ đứng sau “he” phải được chia ở ngôi thứ 3 số ít, không phải ở dạng nguyên mẫu”. Như vậy, nếu từ “he” được tìm thấy và động từ theo sau nó không ở ngôi thứ 3 số ít, ta phát hiện được lỗi ngữ pháp trong câu [5, 18].
- **Statistical approach:** phương pháp thứ ba thường dùng để kiểm tra ngữ pháp là phương pháp thống kê. Ý tưởng chính của phương pháp này là một đoạn văn bản được xem là đúng ngữ pháp nếu các thành phần trong đoạn văn bản đó được sử dụng trong nhiều văn bản khác. Do đó, ta có thể thu thập và thống kê các đoạn văn bản và lưu trữ lại để phục vụ cho mục đích phát hiện lỗi ngữ pháp. Khi sử dụng phương pháp thống kê, có hai cách khác nhau để phát hiện và sửa lỗi ngữ pháp. Một cách sử dụng dữ liệu có sẵn để so sánh trực tiếp với đoạn văn bản cần kiểm tra. Một cách khác là khai thác các cấu trúc ngữ pháp thu thập được từ dữ liệu đã thống kê, sau đó kiểm tra dựa trên dữ liệu này [13, 11].

Chương 2. TỔNG QUAN

2.2.2 Ứng dụng của bài toán kiểm tra ngữ pháp

Bài toán kiểm tra ngữ pháp được áp dụng trong nhiều ứng dụng khác nhau [11], bao gồm:

- Phát hiện lỗi ngữ pháp của văn bản là một trong những ứng dụng cơ bản nhất của bài toán grammar checker. Tính năng kiểm tra lỗi thường được cài đặt trong các phần mềm xử lý văn bản như Microsoft Office hay Open Office.
- Bài tập trắc nghiệm điền khuyết là các ứng dụng cho phép nhập vào một câu có chứa chỗ trống và các phương án trắc nghiệm là các từ thích hợp có thể điền vào chỗ trống đó. Từ các dữ liệu nhập vào, chương trình sẽ lần lượt thế các phương án vào chỗ trống, từ đó tìm ra phương án được cho là thích hợp nhất trả về cho người dùng.
- Bài tập xác định hình thái từ. Đây là dạng ứng dụng tương tự như bài tập trắc nghiệm điền khuyết, tuy nhiên ở đây chương trình sẽ phải tìm ra hình thái chính xác của một từ chỉ định trước khi xét trong ngữ cảnh cụ thể. Ví dụ như bài tập xác định động từ nên được chia ở thì hiện tại đơn, quá khứ đơn, quá khứ hoàn thành, ... Để giải được dạng bài tập này ta cần thêm một cơ sở dữ liệu về hình thái của các từ.
- Bài tập điền từ vào chỗ trống. Tương tự như bài tập trắc nghiệm điền khuyết, tuy nhiên nội dung nhập vào chỉ là một câu có chứa chỗ trống, nhiệm vụ của chương trình là tìm ra từ thích hợp để điền vào chỗ trống đó.

2.3 Khảo sát, các công trình liên quan

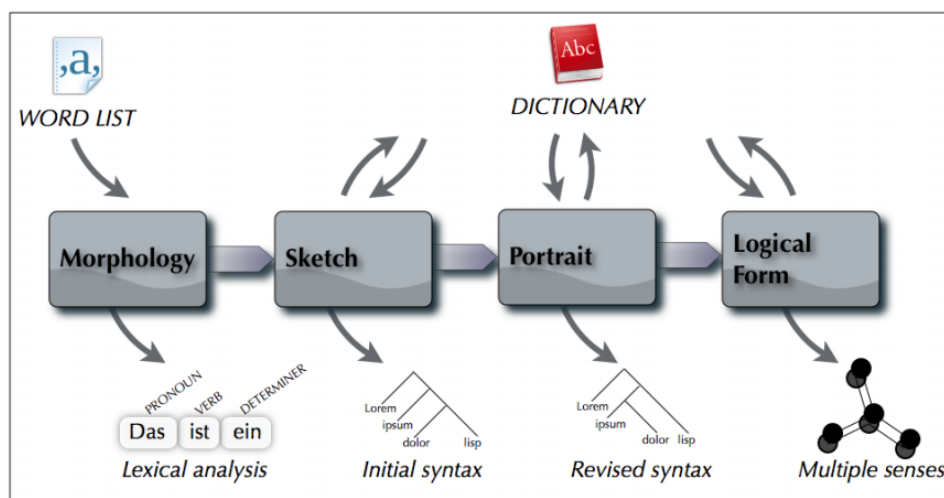
2.3.1 Phương pháp hệ luật dẫn

Các hệ thống kiểm tra ngữ pháp sử dụng hệ luật dẫn bắt đầu được phát triển từ những năm 1980 và đã có nhiều công trình được công bố như MacDonald (1983), Heidorn (1988). Cách tiếp cận này tiếp tục được nghiên cứu sử dụng đến những năm 2000 (Arppe, 2000; Johannessen, 2002;...) và là phương pháp chính để kiểm tra ngữ pháp trong phần mềm Microsoft Word 97, WordPerfect, hay LanguageTool và Grammarian Pro X [5]. Các ứng dụng này đều có phiên bản hoạt động trên nhiều ngôn ngữ khác nhau. Điểm chung của chúng là một số lượng nhất định các luật được áp dụng để kiểm tra ngữ pháp của văn bản. Chính vì sử dụng luật nên điểm yếu của chúng là chỉ hoạt động trên các ngôn ngữ đã được đưa ra luật. Các luật có thể được áp dụng trên nhiều hơn một ngôn ngữ nhưng rất hiếm khi xảy ra.

Microsoft Word 97 Grammar Checker

Quá trình kiểm tra ngữ pháp trong Microsoft Word 97 được tách ra thành nhiều bước như hình 2.1

Chương 2. TỔNG QUAN



Hình 2.1: Hệ thống NLP của Microsoft

1. Bước đầu tiên trong quy trình xử lý ngôn ngữ tự nhiên của Microsoft là bước phân tích từ ngữ. Đoạn văn bản được tách ra thành các token, chủ yếu là các từ và dấu câu. Mỗi token sau đó được phân tích hình vị và lưu thành một entry trong bộ nhớ sử dụng cấu trúc từ điển. Các cụm giới từ, ví dụ như in front of sẽ được xử lý theo cách khác. Thay vì lưu từng từ, hệ thống sẽ lưu cả cụm từ vào dictionary. Tại bước này có thêm hai loại token được quan tâm: factoids và captoids. Factoids thường là các từ về ngày tháng và địa chỉ còn captoid là các từ được in hoa. Kết quả của quá trình này là một dictionary có thành phần là các entry đã được gán nhãn ngữ nghĩa.
2. Bước thứ hai trong quy trình là phân tích. Các luật tăng cường cấu trúc ngữ pháp cụm từ (Augmented phrase structure grammar – APSG) được áp dụng để xây dựng cây dẫn xuất. Kết quả của bước này là một hoặc nhiều cây. Mỗi node trên cây tương ứng với một bảng ghi chứa cặp giá trị - thuộc tính dùng để mô tả đoạn văn bản đang xử lý.
3. Bước thứ ba là bước tối ưu kết quả từ bước thứ hai. Quá trình này đưa ra các đỉnh kèm thích hợp cho các bổ ngữ.
4. Bước cuối cùng trong quá trình sẽ đưa ra các trạng thái logical để làm rõ ngữ nghĩa của đoạn văn bản.

2.3.2 Phương pháp thống kê

Năm 2006, Alam công bố hệ thống kiểm tra ngữ pháp cho tiếng Bangla và tiếng Anh sử dụng phương pháp thống kê theo n-gram. Hệ thống này dựa trên các phân tích thống kê từ các từ vựng và nhãn từ loại để quyết định câu là có ngữ pháp đúng hay sai [8]. Sharma và Jaiswal (2010) phát triển mô hình giúp giảm tỉ lệ lỗi trong quá trình dịch thuật từ tiếng Ấn Độ sang

Chương 2. TỔNG QUAN

Bigrams	Trigrams	Tetragrams
<NNS,VBD>	<NNS,VBD,RB>	<NNS,VBD,RB,RB>
<VBD,RB>	<VBD,RB,RB>	<VBD,RB,RB,JJ>
<RB,RB>	<RB,RB,JJ>	<RB,RB,JJ,IN>
...

Bảng 2.2: Ví dụ các n-grams nhãn từ loại

Chunk type	Description	Example
NC	Noun chunk	A young boy, the girls
VC	Verb chunk	is playing, goes, went
AC	Adjective chunk	more beautiful, younger, old
RC	Adverb Chunk	usually, quickly
PTC	Particle Chunk	up, down
PPC	Prepositional Chunk	at, on, in, under
COC	Conjunction Chunk	and, or, but
QC	Question Chunk	Where, Who, When
INFC	Infinitive Chunk	To
TC	Time Chunk	tomorrow, yesterday

Bảng 2.3: Các loại cụm từ

tiếng Anh. Trong mô hình này họ sử dụng các nguồn ngữ liệu chủ yếu từ các lĩnh vực như du lịch và sức khỏe. Ngoài ra còn có một số công trình khác, ví dụ như Knight và Chandler (1994) hay Han (2006) đã chứng minh được phương án tiếp cận bằng thống kê cho kết quả tốt hơn so với phương pháp dựa trên luật dẫn trong lĩnh vực phát hiện lỗi ngữ pháp [4].

Với phương pháp thống kê theo n-grams, có thể chọn n-grams là các nhãn từ loại hoặc là các token của văn bản. Trong một công trình nghiên cứu do Akshat Kumar công bố năm 2007 đã thống kê các n-gram với đơn vị nhỏ nhất là nhãn từ loại Penn TreeBank cho hệ thống kiểm tra ngữ pháp tiếng Anh dựa trên nguyên tắc hệ miễn dịch nhân tạo [7].

Ví dụ với câu tiếng Anh đã được gán nhãn từ loại: “*Officials/NNS were/VBD not/RB immediately/RB available/JJ for/IN comment/NN*”, ta sẽ thu thập được các n-gram trong bảng 2.2

Một số công trình khác sau khi gán nhãn từ loại xong sẽ tiến hành gom thành cụm từ loại (cụm danh từ, cụm động từ...) và phân tích thành các n-gram là chuỗi các cụm từ này [8]. Các cụm từ thường dùng được trình bày trong bảng 2.3

Ngoài ra còn có phương pháp chọn n-gram với đơn vị chính là các từ có trong đoạn ngữ liệu mà không qua bước xử lý gán nhãn ngữ nghĩa. Rogelio Nazar và đồng nghiệp đã sử dụng phương pháp này để đề xuất cách giải quyết cho vấn đề kiểm tra ngữ pháp, giải bài tập trắc nghiệm, bài tập điền vào chỗ trống cho tiếng Anh [11].

2.4 Tổng kết chương

Các hướng tiếp cận như pattern matching hay rule-based đều đòi hỏi nguồn tri thức từ con người và cần rất nhiều thời gian và chi phí để xây dựng. Hướng tiếp cận bằng thống kê giúp khắc phục các điểm yếu vừa kể ra của hai phương pháp trên. Tuy nhiên phần lớn các hệ thống áp dụng phương pháp này vẫn chưa rút trích ra các đặc điểm từ ngữ pháp của ngôn ngữ. Từ đó, chúng tôi đề xuất một phương pháp thống kê có cải tiến được trình bày ở chương 3.

Chương 3

MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ

3.1 Ý tưởng chính

Hệ thống do chúng tôi phát triển xuất phát từ ý tưởng ứng dụng bài toán kiểm tra ngữ pháp dựa trên thống kê để giải các câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết. Theo đó, khi người dùng nhập vào một câu tiếng Anh có chỗ trống và các đáp án có thể điền vào chỗ trống đó, chương trình sẽ lần lượt thế các đáp án vào vị trí chỗ trống, trích xuất các n-grams liên quan và lấy tổng tất cả các tần suất xuất hiện của n-grams này. Phương án nào cho giá trị lớn nhất được xem là đáp án đúng và trả về kết quả cho người dùng.

Phương pháp phát hiện lỗi dựa trên thống kê nên cần một lượng lớn dữ liệu text mẫu để huấn luyện. Các nội dung text này sẽ được trải qua các công đoạn xử lý như tách câu, tách token, gán nhãn từ loại và trích xuất ra các n-grams. Sản phẩm thu được từ các công đoạn xử lý trên sẽ được lưu trữ trên cơ sở dữ liệu cùng với thông tin về số lần xuất hiện n-grams đang xét trong suốt quá trình huấn luyện.

3.2 Thống kê n-grams kết hợp gom nhóm chủ ngữ

Trong ngữ pháp tiếng Anh, đối với chủ ngữ là ngôi thứ nhất, ngôi thứ 2 và ngôi thứ 3 số nhiều, ở thì hiện tại đơn ta chia động từ ở dạng nguyên mẫu. Ví dụ:

- I have to go to school
- You have to go to school
- They have to go to school

Chương 3. MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ

PRP (personal pronoun)	I	FSP
PRP	He, she, it	TSP
PRP	You, we, they	PP
NNP (proper noun, singular)	John	TSP
NNPS (proper noun, plural)	Vikings	PPS

Bảng 3.1: Phân loại các nhóm chủ ngữ

English sentence	He lives in Ho Chi Minh city
Semantic assigned sentence	He/PRP lives/VBZ in/IN Ho/NNP Chi/NNP Minh/NNP city/NN
Bigram	TSP lives; lives in; in TSP; TSP city
Trigram	TSP lives in; lives in TSP; in TSP city

Bảng 3.2: Ví dụ phân loại nhóm chủ ngữ cho câu

Tuy nhiên, đối với động từ *to be* lại có sự tương đồng giữa ngôi thứ nhất số nhiều, ngôi thứ hai và ngôi thứ ba số nhiều. Tương tự, đối với danh từ riêng hoặc các chủ ngữ thuộc ngôi thứ ba số ít sẽ có cách chia khác nhau. Từ những nhận xét trên, chúng tôi chia chủ ngữ thành ba nhóm và đặt tên như sau:

- Ngôi thứ nhất số ít – First singular pronoun - FSP
- Chủ ngữ số nhiều (We, they, you, danh từ riêng số nhiều) – Plural pronoun - PP
- Chủ ngữ số ít (he, she, it, danh từ riêng số ít) – Third singular pronoun – TSP

Dựa vào các nhãn từ loại của Penn Treebank ta sẽ tiến hành phân nhóm và thay thế tên nhóm vào vị trí của từ đang xét theo như bảng 3.1

Để giảm bớt độ phức tạp nên ta sẽ bỏ qua, không phân nhóm chủ ngữ cho các trường hợp chủ ngữ là các từ có nhãn NN (danh từ số ít) và NNS (danh từ số nhiều) vì dễ bị trùng với trường hợp danh từ làm tân ngữ, bổ ngữ trong câu.

Đối với chương trình giải câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết, bên cạnh các câu hỏi kiểm tra ngữ pháp, chia động từ còn có các dạng về chọn cụm từ phù hợp với ngữ nghĩa, thành ngữ... Do đó ta chỉ xử lý các nhãn ngữ nghĩa được liệt kê ở bảng trên, các thành phần còn lại vẫn giữ nguyên và tiến hành thu thập n-grams. Trong ví dụ ở bảng 3.2, “Ho Chi Minh” là danh từ riêng và được gán thành ba nhãn NNP đứng gần nhau nên ta sẽ gom lại chỉ còn 1, do đó tổng cộng ta có 4 bigrams và 3 trigrams.

Việc gom nhóm chủ ngữ này giúp tăng tần số xuất hiện của các trường hợp tương đồng, giảm bớt sự phân tán tần số xuất hiện không đáng có cho các chủ ngữ khác nhau nhưng cùng ngữ pháp chia động từ. Ở bước kiểm tra so sánh để tìm ra đáp án ta cũng thực hiện việc gom nhóm chủ ngữ tương tự, nhờ đó với n-grams rơi vào các trường hợp chung sẽ cho ra kết quả chính xác hơn.

Chương 3. MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ

Hội thoại			
Tên	Thể loại	Số files	Số từ
charlotte	Mặt đối mặt	93	198,295
switchboard	Điện thoại	2,307	3,019,477
Tổng cộng		2,410	3,217,772
Hội thoại			
Tên	Thể loại	Số files	Số từ
911 report	government, technical	17	281,093
berlitz	travel guides	179	1,012,496
biomed	technical	837	3349714
eggan	fiction	1	61746
icic	letters	245	91318
oup	non-fiction	45	330524
plos	technical	252	409280
slate	journal	4531	4238808
verbatim	journal	32	582384
web data	government	285	1048792
Tổng cộng		6424	11406155
Tổng tất cả		8832	14623927

Bảng 3.3: Thống kê dữ liệu trong OANC

3.3 Thu thập dữ liệu

Bộ ngữ liệu được chúng tôi sử dụng trong chương trình là bộ ngữ liệu OANC (Open American National Corpus) được cung cấp miễn phí tại [trang chủ của OANC](#). Tổng hợp nội dung của bộ ngữ liệu này được thể hiện ở bảng 3.3

Mỗi chủ đề trong OANC được tổ chức thành một folder gồm nhiều file, cấu trúc và nội dung của các file này được mô tả trong bảng 3.4. Các file xml được cấu trúc theo XCES schema. Dữ liệu trong các file xml này được thể hiện dựa trên Standoff markup.

Từ bộ ngữ liệu này, tiến hành thu thập các n-grams, nhãn từ loại và tần suất của chúng dựa trên các file -hepple.xml, -s.xml và file văn bản của chủ đề, lưu trữ trong cơ sở dữ liệu để tiện cho quá trình xây dựng hệ thống sau này cũng như tốc độ truy xuất. Nội dung của các file -hepple.xml và -s.xml được mô tả ở hình 3.1 và hình 3.2.

Chương 3. MÔ HÌNH KIỂM TRA NGŨ PHÁP DỰA TRÊN THỐNG KÊ

filename.anc	Các thông tin chi tiết của chủ đề (tên, nguồn gốc, mô tả...) và định nghĩa các file liên quan trong cùng folder.
filename.txt	Toàn bộ nội dung văn bản của chủ đề.
filename-logical.xml	Thể hiện cấu trúc logic của nội dung văn bản (ghi chú vị trí các đoạn văn, chú thích...)
filename-s.xml	Lưu trữ thông tin các câu có trong nội dung văn bản.
filename-hepple.xml	Các nhãn từ loại Penn Treebank trong nội dung văn bản.
filename-biber.xml	Các nhãn từ loại Biber trong nội dung văn bản.
filename-np.xml	Các cụm danh từ có trong văn bản.
filename-vp.xml	Các cụm động từ có trong văn bản.

Bảng 3.4: Cấu trúc tổ chức file của một chủ đề trong OANC

```
<?xml version="1.0" encoding="UTF-8"?>
<cesAna xmlns="http://www.xces.org/schema/2003" version="1.0.4">
  <struct type="tok" from="24" to="36">
    <feat name="base" value="introduction"/>
    <feat name="msd" value="NNP"/>
  </struct>
  <struct type="tok" from="45" to="50">
    <feat name="base" value="older"/>
    <feat name="msd" value="JJR"/>
  </struct>
  <struct type="tok" from="51" to="57">
    <feat name="msd" value="NNS"/>
    <feat name="base" value="adult"/>
    <feat name="affix" value="s"/>
  </struct>
  ...
</cesAna>
```

Hình 3.1: Cấu trúc một file -hepper.xml

Chương 3. MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ

```
<?xml version="1.0" encoding="UTF-8"?>
<cesAna xmlns="http://www.xces.org/schema/2003" version="1.0.4">
  <struct type="s" from="24" to="36"/>
  <struct type="s" from="45" to="228">
    <feat name="id" value="p1s1"/>
  </struct>
  <struct type="s" from="237" to="655">
    <feat name="id" value="p1s2"/>
  </struct>
  <struct type="s" from="656" to="735">
    <feat name="id" value="p1s3"/>
  </struct>
  <struct type="s" from="744" to="827">
    <feat name="id" value="p2s1"/>
  </struct>
  ...
</cesAna>
```

Hình 3.2: Cấu trúc một file –s.xml

Để trích xuất n-grams từ các chủ đề trong bộ ngữ liệu OANC, chúng tôi đã xây dựng một giải thuật gồm nhiều hàm và được mô tả bằng pseudo-code như sau:

- Hàm *GetSentences*: dùng để lấy tất cả các câu có trong file văn bản của chủ đề đang xét dựa vào nội dung file –s.xml. Tham số truyền vào gồm tên của chủ đề và thư mục hiện tại chứa các file của chủ đề đang xét (hình 3.3).
- Hàm *ProcessSentence*: dùng để tiền xử lý các câu có được từ kết quả trả về của hàm *GetSentences* (hình 3.4)
- Hàm *GetGrams*: đưa ra các n-grams tùy thuộc vào giá trị truyền vào của tham số n. Tham số còn lại của hàm là đoạn văn có được từ kết quả trả về của hàm *ProcessSentence* (hình 3.5)

Chương 3. MÔ HÌNH KIỂM TRA NGŨ PHÁP DỰA TRÊN THỐNG KÊ

```
GetSentences(baseName, workingDirectory)
{
    sentenceBoundaryFile = workingDirectory + '/' + baseName + '-s.xml'
    sentenceBoundaryContent = ReadXmlFile(sentenceBoundaryFile)

    sentences = []

    foreach(xmlNode in sentenceBoundaryContent.AllNodes)
    {
        fromIndex = xmlNode.GetAttributeValue('from')
        toIndex = xmlNode.GetAttributeValue('to')
        sentence = textFileContent.SubString(fromIndex, toIndex)
        sentences.Add(sentence)
    }

    return sentences;
}
```

Hình 3.3: Hàm GetSentences

Chương 3. MÔ HÌNH KIỂM TRA NGŨ PHÁP DỰA TRÊN THỐNG KÊ

```
ProcessSentences(sentence)
{
    removedPatterns = new[]
    {
        @"(?<Month>\d{1,2})/(?<Day>\d{1,2})/(?<Year>(?:\d{4}|\d{2}))",
        @"(?<Month>\d{1,2})-(?<Day>\d{1,2})-(?<Year>(?:\d{4}|\d{2}))",
        @"(?<Protocol>\w+):\/\/(?<Domain>[\w@][\w.:@]+)\/?[\w\.\?=%&=\\-@/$,]*",
        @"([a-zA-Z0-9_\\-\\.]+)@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.|((([a-zA-Z0-9\\-]+\\.)+))([a-zA-Z]{2,4}|[0-9]{1,3}))",
        @"\\d"
    };

    splitCharacters =
    {
        ',', '+', '\\n', '\\r', '(', ')', '{', '}', '[', ']', '&', '"', "'", '?', '!', '!', '<', '>', '-', ':',
    };

    foreach(pattern in removePatterns)
    {
        // Sử dụng Regular Expression để loại bỏ
        // các nội dung không cần thiết trong câu
        sentence = sentence.Replace(pattern, "")
    }

    parts = sentence.Split(splitCharacters)

    return parts
}
```

Hình 3.4: Hàm ProcessSentences

Chương 3. MÔ HÌNH KIỂM TRA NGŨ PHÁP DỰA TRÊN THỐNG KÊ

```
GetGrams(part, n)
{
    words = part.Split(' ')
    bigrams = []
    tokens = []
    if (words.Length < 2)
        continue
    foreach(word in words)
    {
        token = GomNhomChuNguTengAnh(word)
        tokens.Add(token)
    }


    for (i = 0 to tokens.Length)
    {
        gram = ""
        for (j = 0 to n)
        {
            index = i + j
            if (index >= tokens.Length)
                break
            gram = gram + " " + token
        }


        grams.Add(gram)
    }

    return grams
}
```

Hình 3.5: Hàm GetGrams

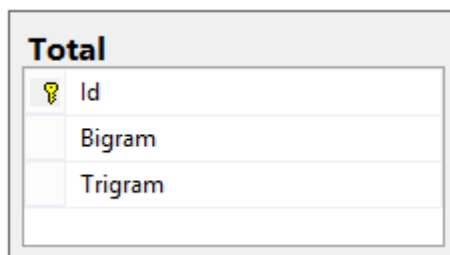
Về phần lưu trữ dữ liệu, chúng tôi sử dụng hệ quản trị cơ sở dữ liệu Microsoft SQL Server 2012. Ứng với mỗi loại n-grams (bigrams và trigrams) sẽ có 26 table, mỗi table tương ứng với một chữ cái trong bảng mẫu tự alphabet. Cấu trúc mỗi bảng đều giống nhau và được mô tả ở hình 3.6. Ngoài ra còn có một bảng Total để lưu trữ tổng số lượng các n-grams thu thập được trong quá trình huấn luyện (hình 3.7).

a3	
	Id
	Gram
	Posibility

a2	
	Id
	Gram
	Posibility

Hình 3.6: Cấu trúc bảng lưu trữ các bigrams và trigrams

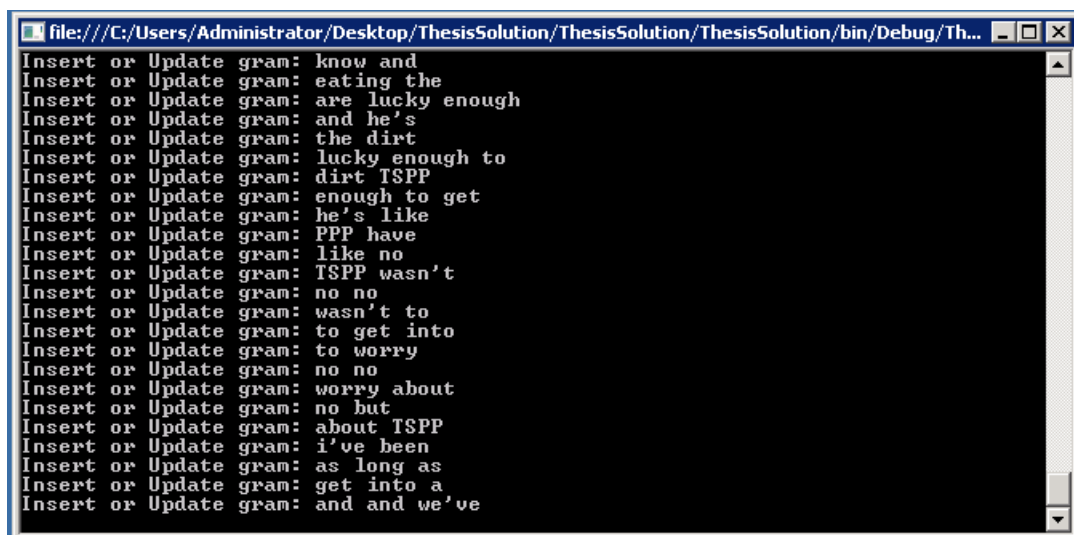
Chương 3. MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ



Id	Bigram	Trigram

Hình 3.7: Bảng Total

Từ những giải thuật cũng như thiết kế cơ sở dữ liệu trên, chúng tôi xây dựng một ứng dụng nhỏ để xử lý dữ liệu từ bộ ngữ liệu OANC và đưa vào lưu trữ trong SQL Server (hình 3.8).



Hình 3.8: Giao diện chương trình xử lý và thu thập dữ liệu từ OANC

3.3.1 Giải thuật chọn đáp án

Hệ thống do chúng tôi đề xuất có thể trả lời câu hỏi trắc nghiệm tiếng Anh điền khuyết tương tự như mẫu câu dưới đây (bảng 3.5).

Để đưa ra được gợi ý đáp án cho câu hỏi trên, các quá trình chủ yếu cần phải thực hiện bao gồm:

- **Tổng hợp dữ liệu:** trước khi so khớp, kiểm tra các n-grams, ta cần thực hiện các công đoạn gán nhãn từ loại cho câu, gom nhóm chủ ngữ như mô tả ở mục 3.1, sau đó tách ra thành các n-grams.

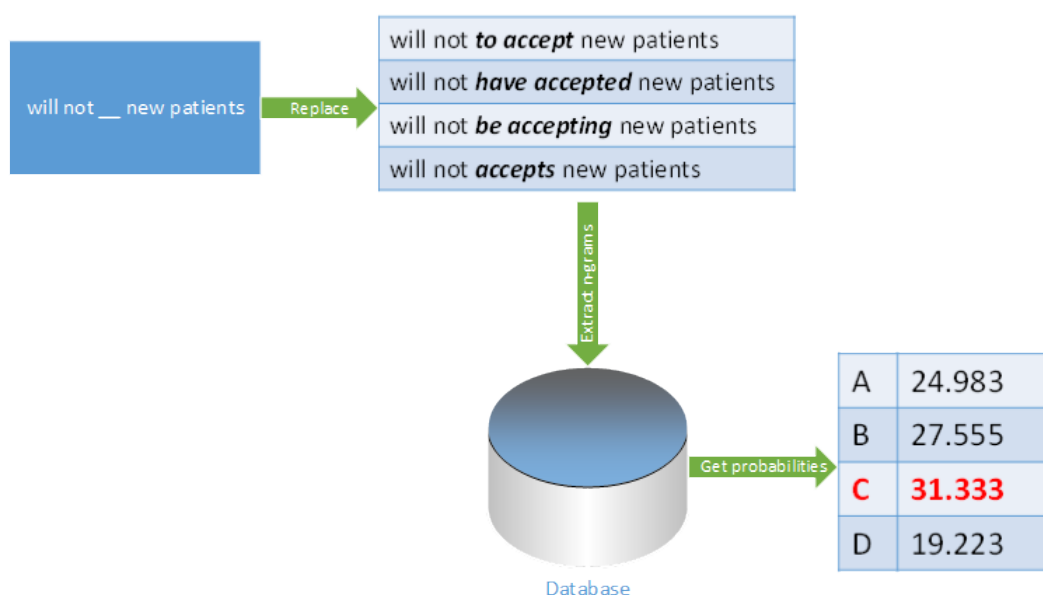
Chương 3. MÔ HÌNH KIỂM TRA NGỮ PHÁP DỰA TRÊN THỐNG KÊ

I'm sorry, but Dr. Klinger and Dr. Moore will not __ new patients at this time.	
A	to accept
B	have accepted
C	be accepting
D	accepts

Bảng 3.5: Ví dụ câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết

- **Kiểm tra tần số n-grams:** sau khi lấy được các n-grams, truy vấn cơ sở dữ liệu để lấy được số lần xuất hiện của các n-grams, sau đó đem chia cho tổng số n-grams có được từ bảng Total. Vì kết quả của phép chia này khá nhỏ nên để tiện so sánh, ta lấy trừ log chúng, sau đó so sánh, lấy phương án có giá trị lớn nhất làm đáp án.

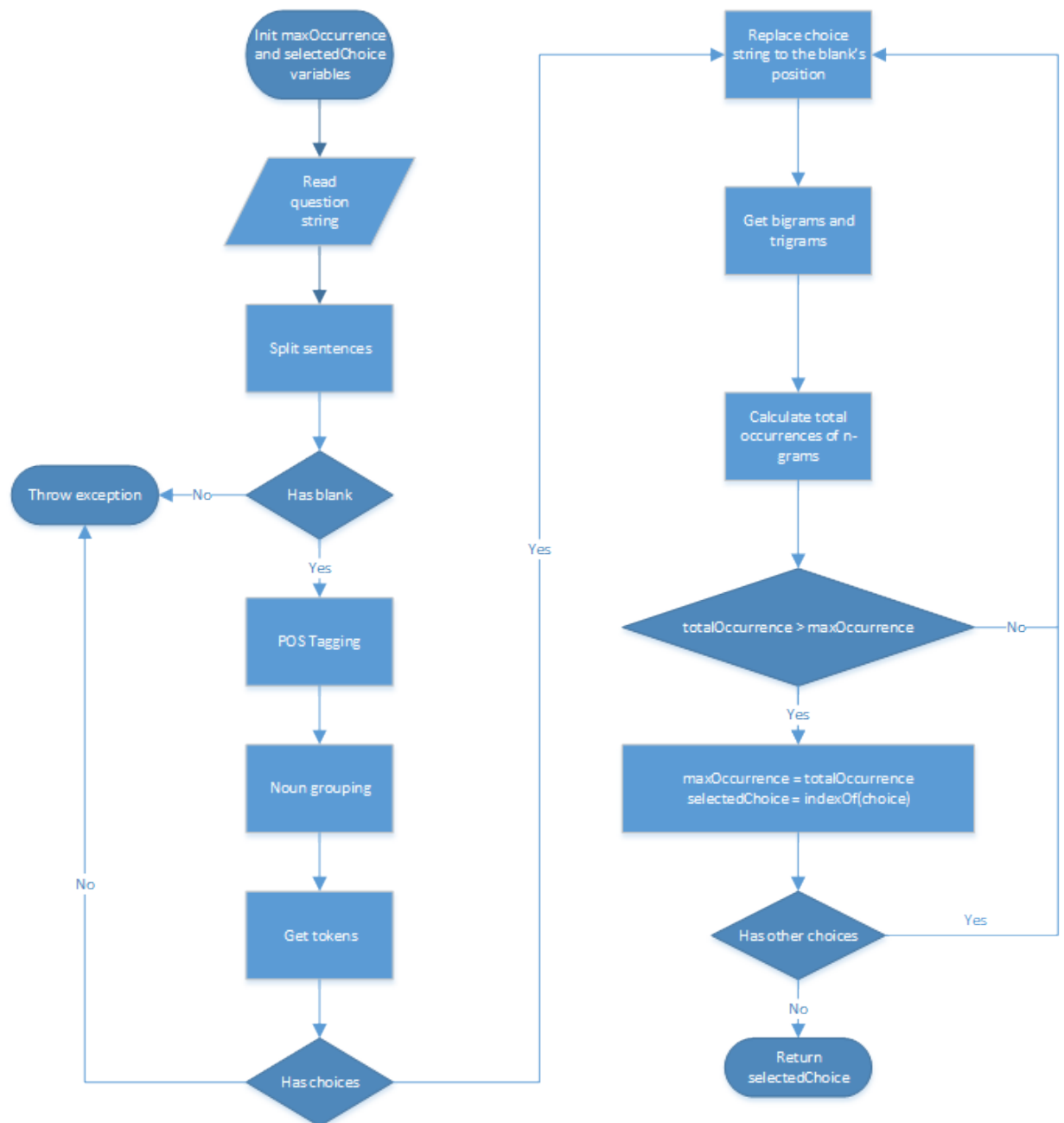
Hình 3.9 thể hiện hình ảnh mô phỏng quá trình hoạt động của hệ thống để đưa ra được đáp án cho câu hỏi.



Hình 3.9: Mô phỏng quá trình đưa ra gợi ý đáp án cho câu hỏi

Hình 3.10 là giải thuật được cài đặt cho hệ thống dựa theo mô phỏng đã mô tả ở trên và được biểu diễn bằng sơ đồ khối.

Chương 3. MÔ HÌNH KIỂM TRA NGŨ PHÁP DỰA TRÊN THỐNG KÊ



Hình 3.10: Sơ đồ khối quy trình lựa chọn phương án cho câu hỏi nhập vào

Để gán nhãn từ loại, chúng tôi sử dụng thư viện gán nhãn *OpenNLP* của Apache, đây là bộ gán nhãn sử dụng giải thuật máy học Maximum Entropy. Ở bước thu thập n-grams, lấy vị trí của chỗ trống làm trung tâm, lần lượt lấy tất cả các n-grams ở xung quanh. Ta có ba phương án thu thập n-grams, mỗi phương án sẽ cho độ hiệu quả khác nhau, gọi *blankIndex* là vị trí của chỗ trống trong câu hỏi, *tokens* là danh sách các từ (word) tách ra từ câu hỏi, bao gồm cả chỗ trống, ta có thể cài đặt 3 phương án như sau:

Chương 3. MÔ HÌNH KIỂM TRA NGŨ PHÁP DỰA TRÊN THỐNG KÊ

- Chỉ lấy các bigrams (Hình 3.11)
- Chỉ lấy các trigrams (Hình 3.12)
- Kết hợp lấy bigrams và trigrams (Hình 3.13)

```
wordsArray = fillString.SplitWords()
for i = blankIndex - 1 to blankIndex + wordsArray.Count
{
    for j = 0 to 2
    {
        if i < 0 || i >= tokens.Count
            continue
        index = i + j
        if index >= tokens.Count
            break
        bigrams.Append(tokens[index] + " ")
    }
    if bigrams.WordsCount > 1
        grams.Add(bigrams)
}
```

Hình 3.11: Phương án chỉ lấy các bigrams

```
wordsArray = fillString.SplitWords()
for i = blankIndex - 2 to wordsArray.Count + 1
{
    if i < 0 || i >= tokens.Count
        continue
    for j = 0 to 3
    {
        int index = j + i;
        if index >= tokens.Count
        {
            break;
        }
        trigrams.Append(tokens[index] + " ");
    }
    if trigrams.Length > 2
        grams.Add(trigrams);
}
```

Hình 3.12: Phương án chỉ lấy các trigrams

```
wordsArray = fillString.SplitWords()

for i = blankIndex - 1 to blankIndex + wordsArray.Count
{
    for j = 0 to 2
    {
        if i < 0 || i >= tokens.Count
            continue
        index = i + j
        if index >= tokens.Count
            break
        bigrams.Append(tokens[index] + " ")
    }
    if bigrams.WordsCount > 1
        grams.Add(bigrams)
}

for i = blankIndex - 2 to wordsArray.Count + 1
{
    if i < 0 || i >= tokens.Count
        continue
    for j = 0 to 3
    {
        int index = j + i;
        if index >= tokens.Count
        {
            break;
        }
        trigrams.Append(tokens[index] + " ");
    }
    if trigrams.WordsCount > 2
        grams.Add(trigrams);
}
```

Hình 3.13: Phương án kết hợp sử dụng bigrams và trigrams

Chương 4

HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

Dựa vào nền tảng về giải thuật, cơ sở dữ liệu đã trình bày ở trên, ta có thể tiến hành xây dựng một hệ thống hoàn chỉnh với các tính năng cơ bản. Mục đích của hệ thống mà chúng tôi xây dựng đó là giúp người dùng (đặc biệt là học sinh, sinh viên) dễ dàng, nhanh chóng có được một gợi ý tin cậy cho câu hỏi tiếng Anh mà họ đang gặp phải. Trong khi đó, smartphone đang phát triển rất mạnh mẽ và xu hướng trong tương lai sẽ trở nên phổ biến. Chính vì những lý do đó nên ta sẽ ưu tiên xây dựng ứng dụng trên di động trước. Môi trường, công cụ phát triển được sử dụng trong khóa luận:

- Hệ điều hành: Window 8 Pro, 64 bit
- IDE: Visual Studio 2012
- .NET Framework 4.0
- Windows Phone 7.5

4.1 Xây dựng Web Service

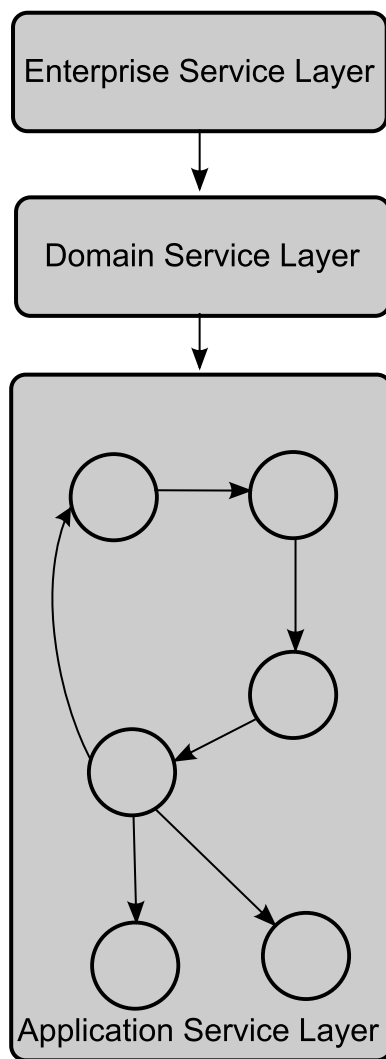
4.1.1 Giới thiệu SOA (Service Oriented Application)

Một trong những xu hướng chính hiện nay trong phát triển phần mềm đó là xây dựng các ứng dụng theo kiến trúc hướng dịch vụ - Service Oriented Architecture (SOA). Mô hình phát triển phần mềm này làm giảm đi sự phụ thuộc của hệ điều hành, dễ dàng và nhanh chóng triển khai trên đa nền tảng, gồm ba lớp chính (hình 4.1) [17]

1. Enterprise Service Layer – cung cấp các dịch vụ cốt lõi như cơ sở dữ liệu, cơ sở tri thức

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

2. Domain Service Layer – cung cấp các dịch vụ theo từng nhóm chức năng, thường gọi là Web Service
3. Application Service Layer – tầng ứng dụng, sử dụng các dịch vụ được cung cấp từ tầng trên để thể hiện các tính năng và tương tác với người dùng.



Hình 4.1: Mô hình kiến trúc SOA

Giao tiếp giữa tầng ứng dụng với web service trong mô hình SOA được thực hiện chủ yếu thông qua các phương thức HTTP như GET, POST, PUT... và nội dung giao tiếp được thể hiện bằng các loại ngôn ngữ đặc tả phổ biến hiện nay như XML hay JSON (Javascript Object Notation), hoàn toàn không phụ thuộc vào hệ điều hành mà web service được triển khai. Do đó bên cạnh khả năng giao tiếp với các ứng dụng client, các web service cũng có thể giao

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

tiếp được với nhau thông qua phương thức này. Ví dụ khi một web service cần lấy thông tin từ web service khác không cùng nằm trên hệ thống hay được triển khai trên hai nền tảng khác nhau (.NET gọi service viết bằng Java và ngược lại). Hiện nay có hai chuẩn xây dựng web service thường dùng nhất:

- **SOAP service:** sử dụng ngôn ngữ đặc tả XML, thông tin trao đổi phải tuân theo cấu trúc của một SOAP Envelope, tương đối phức tạp.
- **RESTful service:** có thể sử dụng XML hoặc JSON làm ngôn ngữ thể hiện thông điệp. Không bắt buộc phải tuân theo một cấu trúc nhất định nào.

Ngày nay RESTful service đang dần được sử dụng phổ biến hơn nhờ vào tính đơn giản, dễ triển khai và tiêu tốn ít tài nguyên mạng trong quá trình trao đổi thông tin.

Các lợi ích của SOA:

- Dễ dàng thực thi thông qua HTTP Method
- Độc lập triển khai, phát triển và quản lý
- Nâng cao khả năng tái sử dụng công nghệ
- Giảm chi phí cho việc cập nhật và bảo trì phần mềm

4.1.2 Tại sao phải sử dụng SOA

Đối với ứng dụng giải câu hỏi trắc nghiệm tiếng Anh điền khuyết mà chúng tôi đề xuất, việc phát triển ứng dụng trên di động là phù hợp nhất nhờ vào tính nhỏ gọn của thiết bị, có thể sử dụng gần như mọi lúc, mọi nơi. Tuy nhiên, đặc trưng của loại thiết bị này là tài nguyên thường ít và hiệu năng tính toán không cao, hơn nữa do sử dụng phương pháp thống kê để giải quyết bài toán nên cần dung lượng đĩa cứng khá lớn. Nếu chỉ đơn thuần xây dựng trên di động sẽ không thể đảm bảo được các yêu cầu về trải nghiệm người dùng, ngoài ra khi muốn triển khai trên các nền tảng khác sẽ rất khó khăn và tốn kém.

Từ đặc điểm cũng như lợi ích của việc xây dựng ứng dụng theo mô hình hướng dịch vụ SOA, ta có thể giải quyết tất cả các vấn đề khó khăn vừa đề cập, bao gồm:

- Quản lý dữ liệu tập trung bằng hệ quản trị cơ sở dữ liệu
- Tốc độ xử lý, truy vấn nhanh chóng
- Dễ dàng cập nhật, bảo trì
- Khả năng mở rộng cho các nền tảng khác cao.

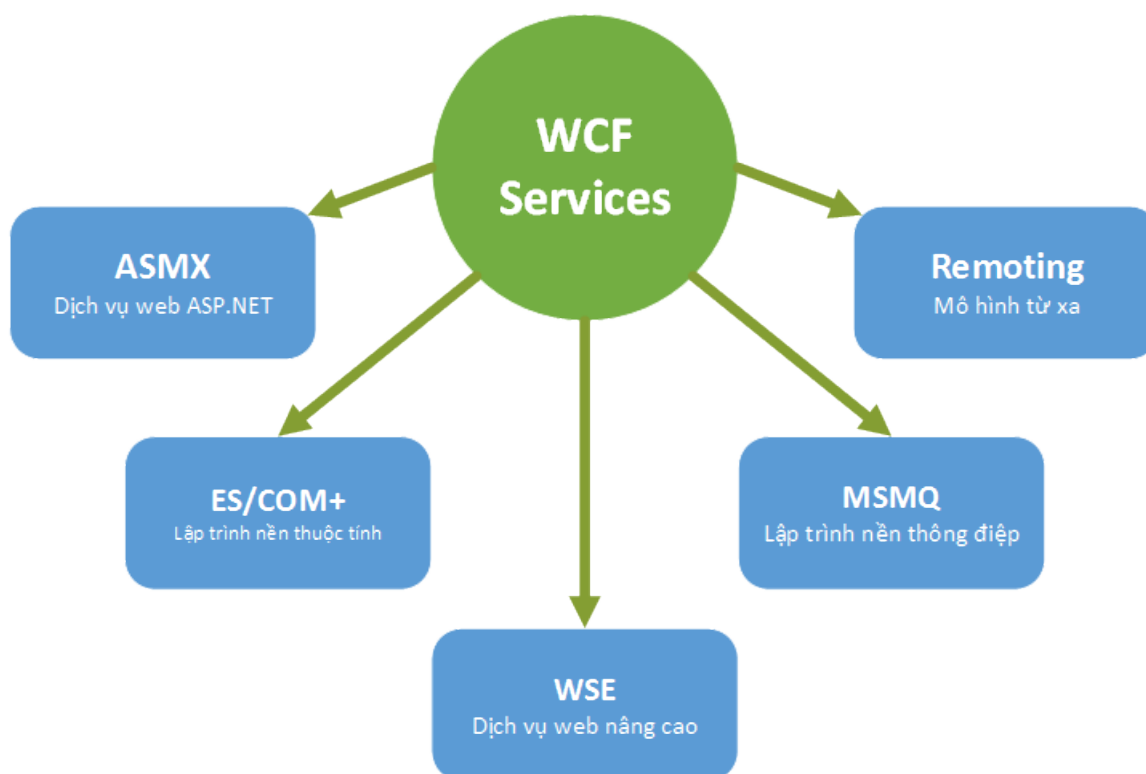
Một trong những công nghệ tốt nhất hiện nay hỗ trợ việc xây dựng ứng dụng theo mô hình SOA là Windows Communication Foundation (WCF). Đây là công nghệ do Microsoft phát triển và hoạt động trên hệ điều hành Windows với .NET Framework. Ở phần tiếp theo ta sẽ tìm hiểu về WCF cũng như cách xây dựng hệ thống trả lời câu hỏi tiếng Anh tự động.

4.1.3 Giới thiệu Windows Communication Foundation

Windows Communication Foundation (WCF) [17] là mô hình lập trình thống nhất của Microsoft cho việc xây dựng các ứng dụng theo hướng dịch vụ (Service Oriented Application). WCF cung cấp các nhà phát triển có một giải pháp xây dựng việc truyền thông an toàn. WCF được xây dựng trên Microsoft .NET Framework. Nó hợp nhất một loạt các hệ thống phân phối trong một kiến trúc composable mở rộng, hỗ trợ vận chuyển nhiều, tin nhắn hình, mã hóa, topo mạng, và các mô hình lưu trữ (hình 4.2). Nó là phiên bản kế tiếp của một số sản phẩm hiện có: ASP.NET 's WebMethod (ASMX), WebService(WSE), .NET Remoting, ...

Với việc ra đời của WCF, mọi phương pháp liên lạc trước kia đều có thể thực hiện trên WCF. Do vậy nhà phát triển chỉ cần làm chủ được công nghệ WCF là có thể xây dựng các ứng dụng một cách nhanh chóng.

Do WCF được xây dựng trên cơ sở của .NET Framework, nó là tập các lớp cho phép các nhà phát triển xây dựng các ứng dụng hướng dịch vụ bằng ngôn ngữ lập trình VB.NET hay C#.



Hình 4.2: Mô hình thống nhất thay thế cho các công nghệ phân tán trước đó

WCF được thiết kế với các nguyên lý sau của SOA

- Rõ ràng về ranh giới: các ứng dụng và dịch vụ liên lạc với nhau thông qua các thông

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

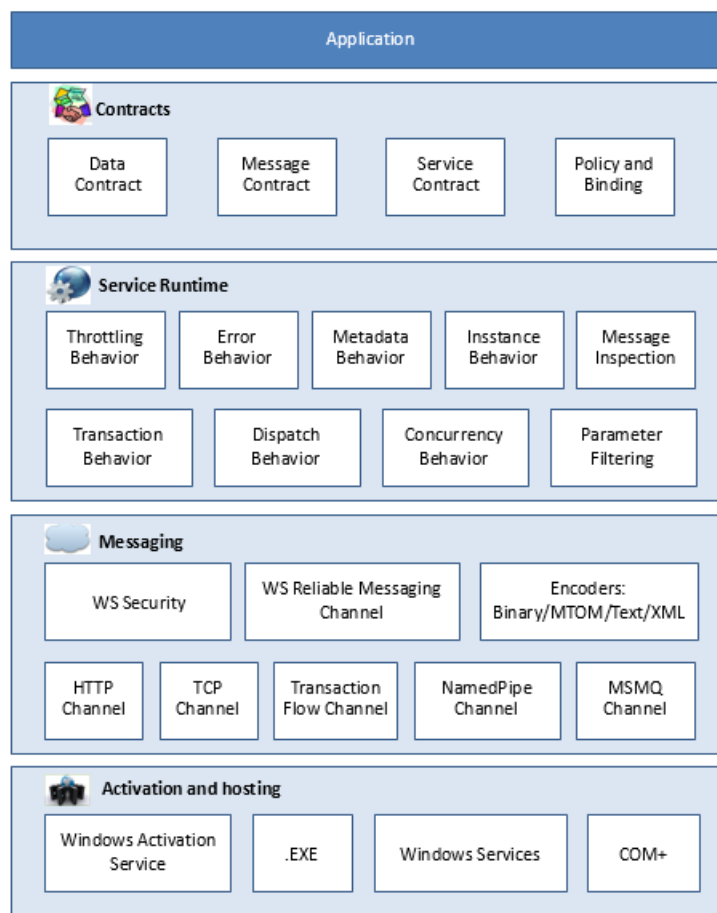
điệp mà không quan tâm đến việc xử lý và tiếp nhận

- Tự phát triển: dịch vụ và sử dụng dịch vụ là độc lập với các quá trình nâng cấp phiên bản, triển khai, hoạt động và bảo mật.
- Chia sẻ yêu cầu, không chia sẻ nội dung: các dịch vụ cung cấp các phép toán và cấu trúc thông tin. Không bao gồm nội dung thông tin.
- Tương thích dựa trên chính sách: các dịch vụ có thể thiết kế để độc lập với việc triển khai, thống nhất với các ứng dụng về chuẩn giao tiếp.

4.1.4 Kiến trúc của WCF

WCF có 4 thành phần chính sau (hình 4.3):

- **Tầng Contract:** Các contract trong WCF cũng chứa các thông tin tương giống như các hợp đồng/hiệp định mà bạn ký trong đời sống thật. Contract định nghĩa các đặc tả trong hệ thống bản tin.
- **Runtime service:** chứa các hành xử sẽ xảy ra trong quá trình thực hiện của dịch vụ, nghĩa là các hành xử thực thi của dịch vụ.
- **Message:** tập hợp các kênh. Mỗi kênh là một thành phần xử lý bản tin theo một cách nào đó. Một tập các kênh thường được gọi là ngăn xếp kênh. Các kênh làm việc trên bản tin và trên đầu đề của bản tin. Lớp này khác với lớp thực thi dịch vụ chủ yếu bởi sự khác nhau trong việc xử lý nội dung bản tin.
- **Host and activation:** nhìn một cách tổng thể thì một dịch vụ thực chất là một chương trình. Cũng giống như các chương trình khác, một dịch vụ cần phải chạy trong một tệp thực thi. Dịch vụ này thường được gọi là dịch vụ tự chứa. Các dịch vụ còn có thể được chứa, hoặc chạy trong một tệp thực thi được quản lý bởi một agent bên ngoài như IIS hay Windows Activation Services (WAS).



Hình 4.3: Kiến trúc của WCF

4.1.5 Xây dựng AnswerService

AnswerService là tên gọi của project WCF được xây dựng với vai trò làm trung tâm xử lý các yêu cầu (request) đến từ phía người dùng (client) và gửi trả về kết quả (response) cho yêu cầu đó. AnswerService sẽ truy cập đến cơ sở dữ liệu SQL Server được mô tả ở trên để lấy các dữ liệu thống kê n-grams. Công nghệ được sử dụng để kết nối giữa WCF Server và SQL Server là Entity Framework, một công nghệ mới thay cho ADO.NET giúp tự động sinh ra các lớp đối tượng tương ứng với các bảng và các thuộc tính của các đối tượng tương ứng với các cột trong mỗi bảng. Entity Framework giúp cho việc truy vấn dữ liệu trong database đơn giản hơn nhờ việc hỗ trợ truy xuất theo kiểu hướng đối tượng đồng thời với kiểu truy vấn bằng query như thông thường.

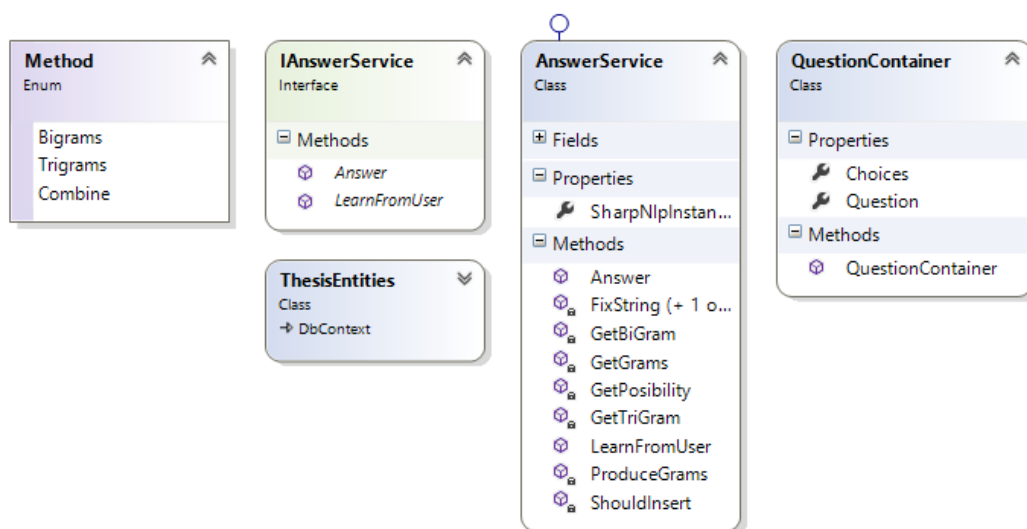
Để xây dựng WCF Service, trước tiên ta cần khởi tạo một Interface định nghĩa cho service (*ServiceContract*) cũng như các phương thức (*OperationContract*) mà service đó cung cấp.

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

Trong hệ thống của chúng tôi IAnswerService (hình 4.4) chính là phần định nghĩa đó. Hai phương thức được định nghĩa trong service bao gồm:

- **Answer**: trả lời cho câu hỏi người dùng nhập vào
- **LearnFromUser**: học thêm n-grams mới từ câu hỏi người dùng nhập vào đáp án do người dùng chỉ định

IAnswerService chỉ là phần định nghĩa cho service cũng như hoạt động mà service cung cấp, do đó ta cần tạo thêm một class AnswerService implement interface này để cài đặt các tính năng đã định nghĩa. Như hình 4.4, ta thấy class AnswerService có 1 method là Answer và các private method khác, tất cả đều được mô tả đầy đủ trong bảng 4.1



Hình 4.4: UML Class trong WCF Service

STT	Tên phương thức	Tham số	Mô tả chức năng
1	FixString	Chuỗi đầu vào Mảng các tokens của chuỗi đầu vào Mảng các nhãn từ loại của từng token	Loại bỏ các khoảng trống thừa. Chuyển đổi chủ ngữ của câu sang nhóm chung tương ứng Thay thế các token đánh giấu là danh từ riêng liên tiếp bằng một nhãn duy nhất (ví dụ: NNP NNP NNP -> NNP)

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

2	GetBigrams	<p>List các tokens đã được gom nhóm chủ ngữ, kể cả khoảng trống cần điền</p> <p>Phương án thay thế vào vị trí chỗ trống</p>	<p>Thế nội dung của phương án thay thế vào chỗ trống.</p> <p>Trả về các bigrams ở xung quanh vị trí chỗ trống.</p>
3	GetTrigrams	<p>List các tokens đã được gom nhóm chủ ngữ, kể cả khoảng trống cần điền</p> <p>Phương án thay thế vào vị trí chỗ trống</p>	<p>Thế nội dung của phương án thay thế vào chỗ trống.</p> <p>Trả về các trigrams ở xung quanh vị trí chỗ trống.</p>
4	GetGrams	<p>Nội dung câu hỏi đầu vào</p> <p>Phương án cần thay thế</p> <p>Một thực thể của lớp SharpNLP</p>	<p>Dùng thể hiện của lớp SharpNLP để gán nhãn từ loại cho câu hỏi đầu vào</p> <p>Gọi hàm FixString để gom nhóm chủ ngữ cũng như xử lý các ký tự không cần thiết trong câu</p> <p>Gọi các hàm GetBigrams và GetTrigrams để lấy ra bigrams, trigrams cần thiết</p>
5	GetPosibility	<p>N-grams cần lấy xác suất</p>	<p>Truy vấn vào cơ sở dữ liệu để lấy tổng số lần xuất hiện của n-grams truyền vào. Để lấy được, ta cần chọn đúng bảng dựa vào ký tự đầu tiên của n-grams và giá trị của n (ví dụ hình 4.5).</p>

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

		Giá trị của n (n = 2 hoặc n = 3)	Dựa vào dữ liệu trong bảng Total, trả về giá trị xác suất của n-grams truyền vào. Nếu không tìm được n-grams trong cơ sở dữ liệu, ta xem như đó là một trường hợp sai ngữ pháp và trả về giá trị -1.
6	Answer	Câu hỏi đầu vào Các phương án lựa chọn để thể vào vị trí chỗ trống	Lần lượt duyệt qua các phương án lựa chọn do người dùng nhập vào. Gọi các phương thức GetGrams để lấy các n-grams cần thiết và GetPosibility cho các n-grams có được. Tính tổng giá trị xác suất của các n-grams khi thể từng đáp án vào. Chọn ra đáp án có tổng xác suất lớn nhất để trả lại kết quả cho người dùng
7	ShouldInsert	Danh sách các n-grams thu được từ hàm GetGrams	Kiểm tra độ tin cậy của đáp án do người dùng chỉ định. Nếu vượt qua ngưỡng yêu cầu mới thêm vào cơ sở dữ liệu
8	ProduceGrams	Câu hoàn chỉnh từ câu hỏi và đáp án chỉ định	Gom nhóm chủ ngữ tiếng Anh và gán nhãn từ loại cho câu Sinh ra tất cả các n-grams có thể có cho câu nhập vào
9	LearnFromUser	Câu hỏi và các phương án trả lời Phương án do người dùng chỉ định	Học từ gợi ý của người dùng với câu hỏi và đáp án do người dùng chỉ định

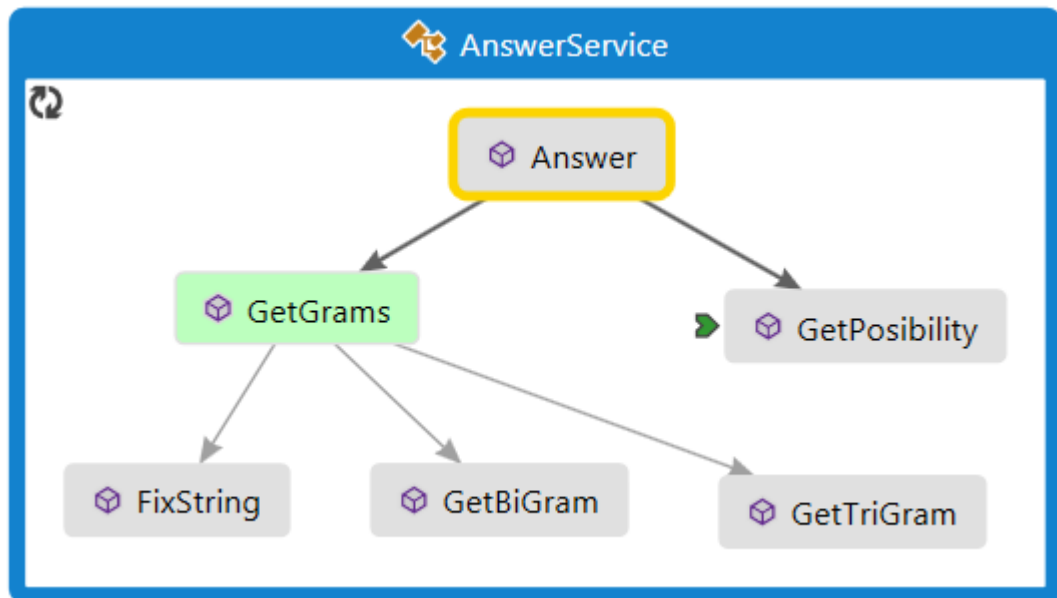
Bảng 4.1: Các phương thức sử dụng trong AnswerService

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

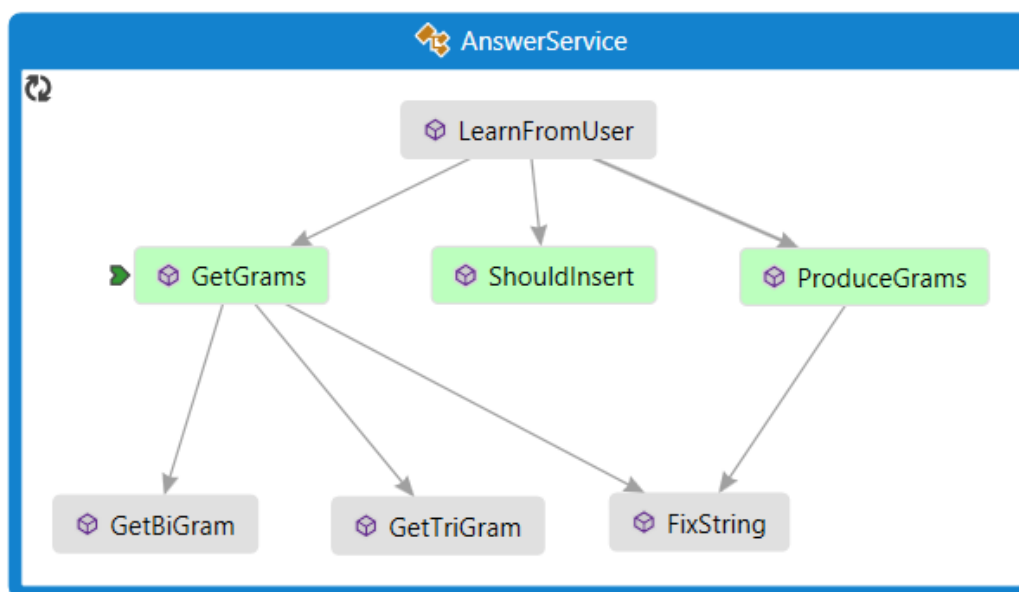
```
using (var db = new ThesisEntities())
{
    string query = string.Format("select Possibility from dbo.{0}{1} where Gram='{2}'",
                                grams[0].ToString().ToLower(), n, grams);
    gramCount = db.Database.SqlQuery<double>(query).FirstOrDefault();
    var tRecord = db.Totals.First();
    total = n == 2 ? tRecord.Bigram : tRecord.Trigram;
}
```

Hình 4.5: Truy vấn cơ sở dữ liệu để lấy tổng số lần xuất hiện của n-grams

Trình tự gọi các phương thức trong AnswerService để có thể trả về đáp án cho người dùng từ câu hỏi và các phương án nhập vào được thể hiện bằng Code map ở hình 4.6. Hình 4.7 thể hiện trình tự thực thi các phương thức khi cần học từ gợi ý của người dùng.



Hình 4.6: Code map thể hiện các phương thức được gọi để trả lời câu hỏi



Hình 4.7: Code map các phương thức được gọi để học từ gọi ý người dùng

Bên cạnh AnswerService, ở hình 4.4 ta còn thấy một vài class khác:

- **Method** là kiểu enum thể hiện các lựa chọn lấy n-grams, bao gồm chỉ lấy bigrams, chỉ lấy trigrams và kết hợp cả trigrams và bigrams như đã mô tả ở phần 3.3.
- **QuestionContainer** là một DataContract, là cấu trúc dữ liệu cho thông điệp sẽ được truyền đi qua lại giữa client và Server. Các thuộc tính của QuestionContainer gồm có:
 - Question: chuỗi câu hỏi có chứa khoảng trống
 - Choices: danh sách các phương án lựa chọn cho câu hỏi
- **ThesisEntities**: là class do Entity Framework sinh ra dựa trên cấu trúc database mà ta sử dụng trong hệ thống.

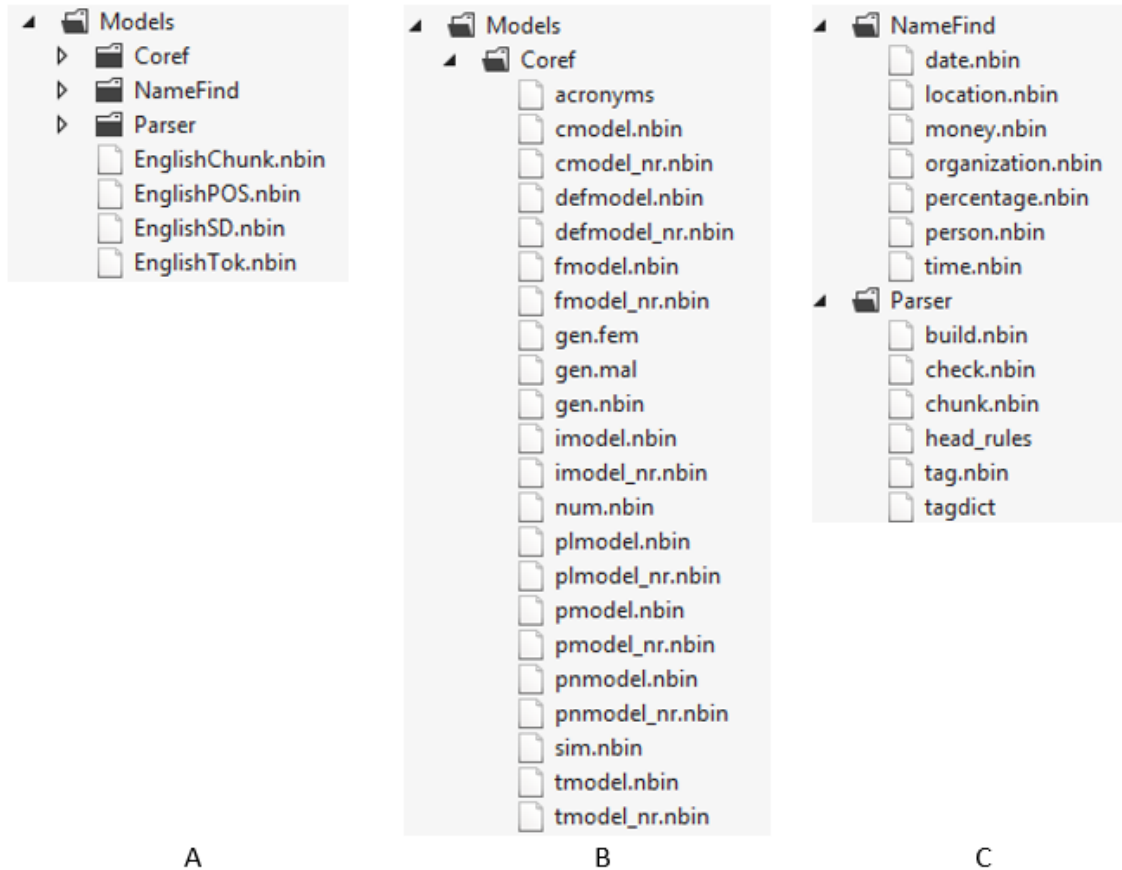
4.1.6 Gán nhãn từ loại bằng SharpNLP

Ở phần trên, trong hàm GetGrams ta có sử dụng một đối tượng thuộc lớp SharpNLP để gán nhãn từ loại. Cụ thể hơn SharpNLP là một thư viện mở rộng của bộ thư viện gán nhãn từ loại OpenNLP của Apache và được cài đặt bằng ngôn ngữ lập trình C#. Bộ thư viện này được cung cấp dưới dạng mã nguồn mở và có thể download miễn phí tại địa chỉ <http://sharpnlp.codeplex.com/>

Để sử dụng bộ gán nhãn từ loại này, trước tiên ta cần chuẩn bị nguồn dữ liệu huấn luyện mẫu có sẵn bằng cách download về các file dữ liệu trên trang chủ của thư viện. Sau khi

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

download xong, các dữ liệu này phải được sắp xếp và tổ chức theo cấu trúc chính xác như hình 4.8.



Hình 4.8: Sắp xếp các file, folder cho dữ liệu huấn luyện của SharpNLP

Sau khi chuẩn bị xong các file cần thiết, thêm các thư viện cần dùng vào project. Ở phần 2.1.6 ta đã biết để gán nhãn từ loại cần thực hiện qua các bước được mô tả như trong hình 4.9



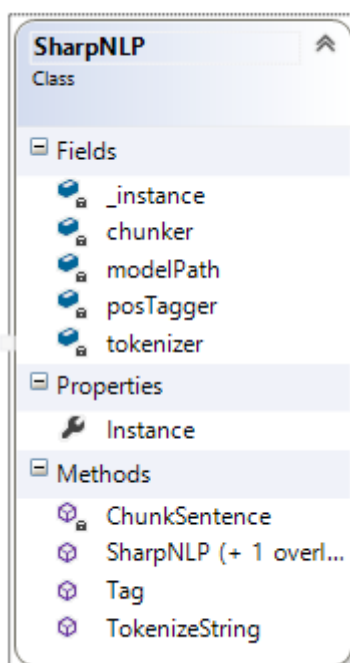
Hình 4.9: Quy trình gán nhãn từ loại

Tương tự như vậy, trong SharpNLP, để gán nhãn từ loại cần khởi tạo hai đối tượng thuộc hai lớp:

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

- `OpenNLP.Tools.Tokenize.EnglishMaximumEntropyTokenizer`: cung cấp phương thức `Tokenize` để từ câu input cho ra tất cả các token của câu.
- `OpenNLP.Tools.PosTagger.EnglishMaximumEntropyPosTagger`: cung cấp phương thức `Tag` để từ mảng các token tách được từ tokenizer, đưa ra mảng gồm các nhãn từ loại của các token.

Để nhanh chóng và tiện lợi hơn, chúng tôi đã xây dựng một lớp `SharpNLP` mới và gom cả hai công đoạn tokenization và tagging vào trong cùng một phương thức. Vì công việc gán nhãn từ loại được dùng xuyên suốt quá trình hoạt động của service nên ta sử dụng design pattern Singleton để truy cập vào đối tượng (hình 4.10).



Hình 4.10: Lớp `SharpNLP`

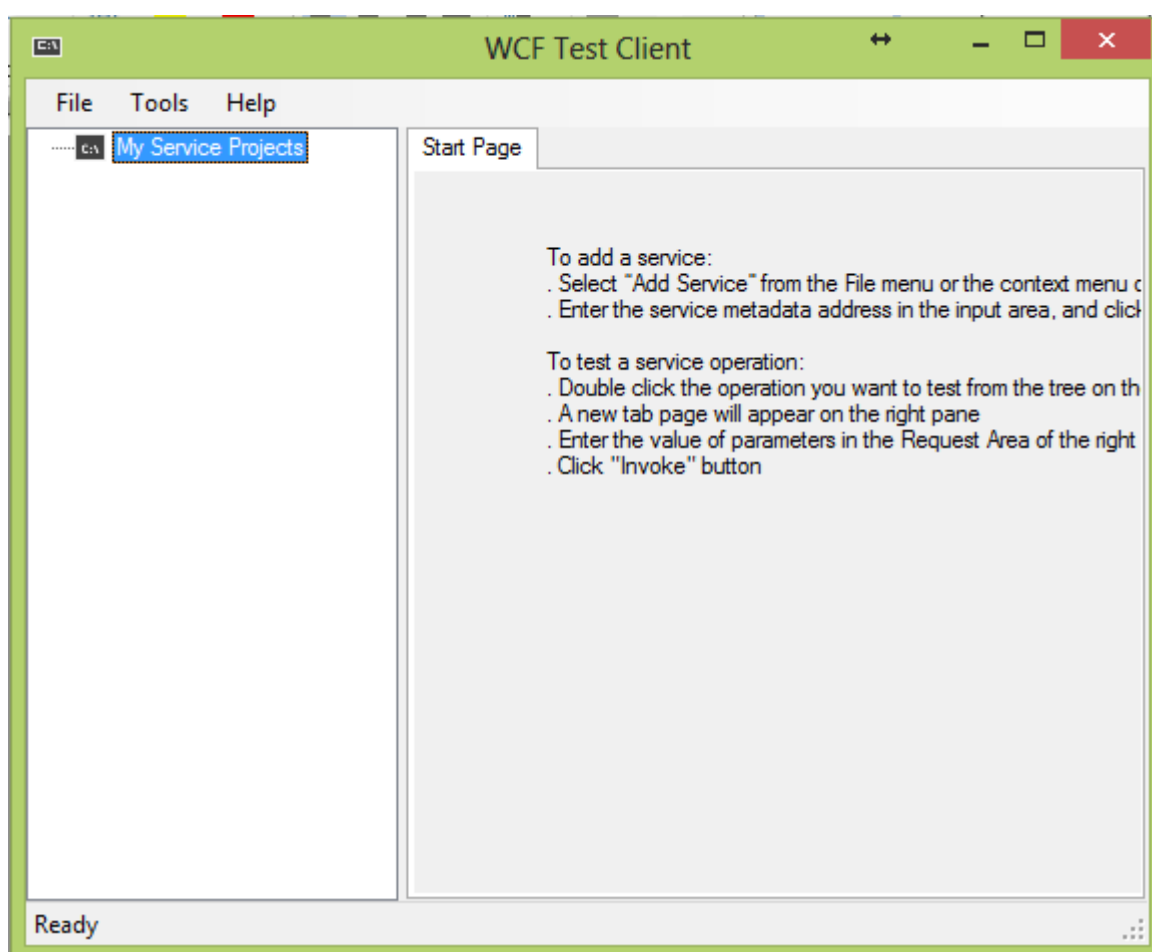
4.1.7 Chạy thử và kiểm tra

Sau khi cài đặt xong các phương thức, ta cần chạy thử và kiểm tra xem WCF service hoạt động có đúng theo yêu cầu hay không. Một web service thường trao đổi thông tin bằng các giao thức HTTP như GET, POST, PUT, DELETE. Nội dung thông điệp được biểu diễn bằng XML hoặc JSON tùy vào loại web service mà người lập trình sử dụng (SOAP Service hoặc RESTful service). Nếu không có công cụ hỗ trợ, để chạy thử web service mà mình vừa viết xong, người lập trình cần thực hiện khá nhiều công đoạn phức tạp, bao gồm:

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

- Deploy web service lên server
- Xây dựng nội dung thông điệp cần kiểm tra theo ngôn ngữ biểu diễn thích hợp
- Gửi request lên server và nhận response.

Nếu thực hiện như trên, công đoạn debug cũng diễn ra khá khó khăn. Với công nghệ xây dựng web service bằng Windows Communication Foundation, Microsoft cung cấp cho người lập trình một bộ công cụ giúp dễ dàng kiểm tra, debug WCF project có tên gọi là WCF Test Client (hình 4.11).



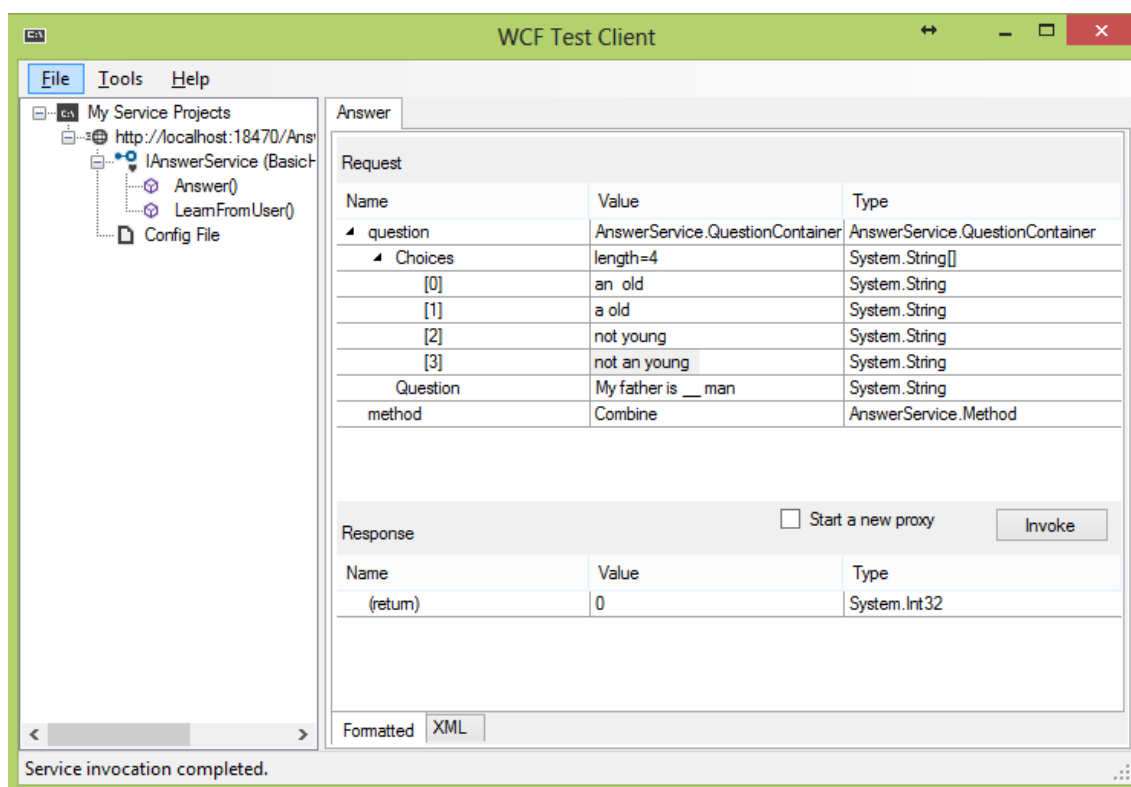
Hình 4.11: Giao diện công cụ WCF Test Client

Để bắt đầu, trước tiên trong cửa sổ ứng dụng Visual Studio, nhấn F5 để Debug (Hoặc Ctrl + F5) để chạy trực tiếp. Visual Studio sẽ tự động deploy WCF Project vào IIS Express – phiên bản thu gọn của IIS (Internet Information Services) của Windows. Nếu cửa sổ WCF Test Client tự động hiện ra, ta chỉ cần chọn vào service nằm dưới mục “My Service

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

Projects” và mở hết tất cả các nhánh con. Các phương thức được định nghĩa trong service sẽ hiện ra cho phép chúng ta chọn, truyền thông điệp và kiểm tra kết quả. Ngược lại, nếu tiến hành Debug mà cửa sổ trình duyệt khởi chạy cùng với một tab có chứa đường dẫn có dạng `http://localhost:xxx` với xxx là số port được cấp ngẫu nhiên lúc ban đầu. Lúc này ta cần phải chạy WCF Test client bằng phương pháp thủ công từ đường dẫn `C:/Program Files/Microsoft Visual Studio 9.0/Common7/IDE/WcfTestClient.exe`. Đường dẫn này có thể thay đổi tùy vào ổ đĩa cài đặt Visual Studio và phiên bản mà người lập trình đang sử dụng. Sau khi khởi chạy chương trình, tiến hành thêm đường dẫn của service trên localhost vào WCF Test Client.

Với AnswerService, double click vào phương thức Answer, ở khung bên phải sẽ cho ta nhập vào nội dung câu hỏi và các đáp án. Sau khi nhập đầy đủ các tham số, nhấn nút Invoke để chương trình gọi service, kết quả trả về được hiển thị ở khung Response (hình 4.12).



Hình 4.12: Test AnswerService với công cụ WCF Test Client

4.2 Sử dụng Hadoop để lấy dữ liệu từ Google Books n-grams

4.2.1 Giới thiệu Hadoop

Hadoop là dự án mã nguồn mở được xây dựng với mục đích cung cấp cơ chế tính toán phân tán trên dữ liệu lớn có độ ổn định, tin cậy cao. Dự án Hadoop được bắt đầu vào năm 2005 bởi Doug Cutting – cũng chính là tác giả của dự án Apache Lucene và Mike Cafarella. Tiền thân của Hadoop là dự án Apache Nutch được bắt đầu vào năm 2002, đây là một phần của dự án Lucene có chức năng là một web search engine. Hadoop đã được đăng ký bản quyền mã nguồn mở theo các điều khoản của giấy phép Apache v2.

Hadoop được viết bằng Java và được thiết kế để có thể hoạt động với cluster có từ một đến hàng ngàn máy tính, mỗi máy tính cung cấp khả năng xử lý và lưu trữ riêng. Thay vì phụ thuộc vào phần cứng để đảm bảo khả năng hoạt động, chiến lược của Hadoop là phát hiện và xử lý các trường hợp lỗi hệ thống ở tầng ứng dụng, nhờ vậy cung cấp dịch vụ có độ ổn định cao [19].

Hadoop bao gồm các module nhỏ sau:

- **Hadoop Common:** các tiện ích chung để hỗ trợ cho các module khác
- **Hadoop Distributed File System (HDFS):** hệ thống dữ liệu phân tán cung cấp khả năng trao đổi, lưu trữ dữ liệu của ứng dụng trong toàn bộ cluster
- **Hadoop YARN:** một framework cho phép định thời công việc và bảo trì tài nguyên của toàn bộ hệ thống.
- **Hadoop MapReduce:** cung cấp cơ chế xử lý song song trên các tập dữ liệu lớn

4.2.2 Giới thiệu HDFS

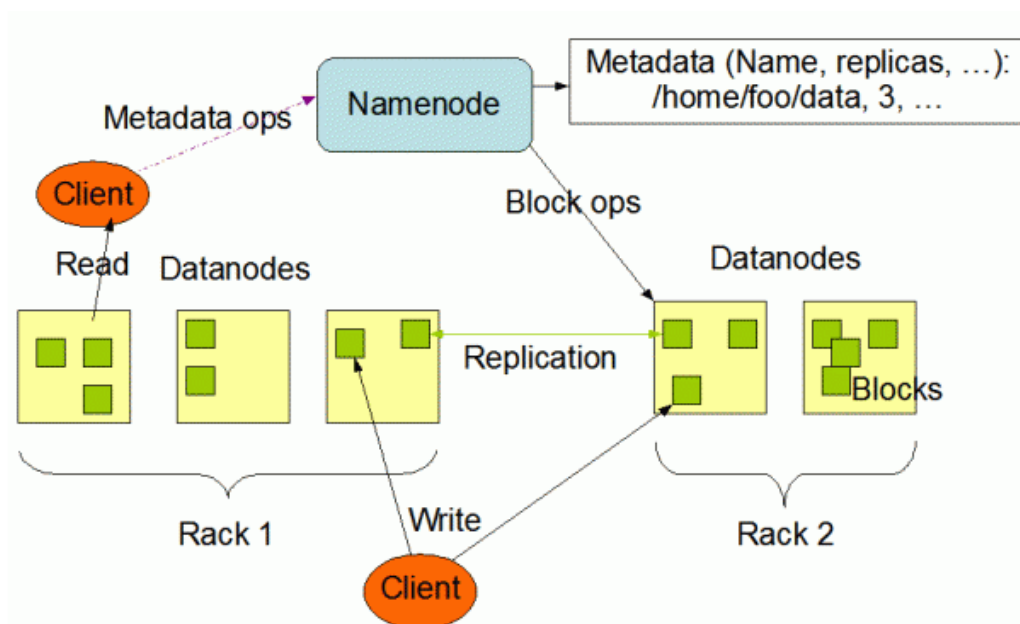
Hadoop Distributed File System (HDFS) là hệ thống quản lý dữ liệu phân tán được thiết kế để có thể hoạt động trên các loại phần cứng phổ biến. Nó có nhiều điểm tương đồng như các mô hình dữ liệu phân tán khác đã có từ trước. Điểm khác biệt của HDFS so với phần còn lại chính là khả năng giảm thiểu lỗi và có thể hoạt động ngay trên cả các phần cứng loại phần cứng thấp, rẻ tiền. HDFS cũng hỗ trợ khả năng streaming dữ liệu. Tuy nhiên, đặc trưng thiết kế của HDFS không phù hợp với các ứng dụng yêu cầu độ trễ trong truyền dẫn dữ liệu thấp.

Các máy trong HDFS cluster gọi là các node. HDFS hoạt động theo cơ chế master-worker và chia ra làm hai loại node:

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

- **Namenode** (đóng vai trò master): đóng vai trò quản lý các namespace của hệ thống dữ liệu. Tất cả thông tin về files, thư mục được lưu trữ bằng cấu trúc cây và quản lý bởi namenode này. Một cluster chỉ có thể có 1 namenode.
- **Datanodes** (là các worker): ngoại trừ namenode, các máy còn lại trong HDFS đều là các datanode. Sau này còn có thêm một node đóng vai trò dự trữ của namenode gọi là secondary namenode.

Trong quá trình hoạt động, mỗi datanode sẽ định kỳ gửi một gói tin về namenode gọi là heartbeat message. Gói tin này đóng vai trò thông báo cho namenode biết rằng datanode đó vẫn còn đang hoạt động được. Sau một khoảng thời gian nhất định mà không nhận được gói tin heartbeat từ một datanode nào đó, namenode sẽ xem datanode đó đã chết và loại trừ datanode đó ra khỏi cluster. Cơ chế giảm thiểu lỗi (fault-tolerant) của HDFS được xây dựng dựa trên các datanode dự phòng gọi là replication. Tùy vào nhu cầu và tài nguyên mà mỗi datanode có số lượng replication khác nhau. Thông thường con số này là 3. Do đó, nếu một datanode bị lỗi, namenode sẽ chuyển sang sử dụng một trong các datanode dự phòng còn lại. Hình 4.13 thể hiện tổng quát các thành phần của HDFS.



Hình 4.13: Kiến trúc HDFS

4.2.3 Xử lý song song với MapReduce

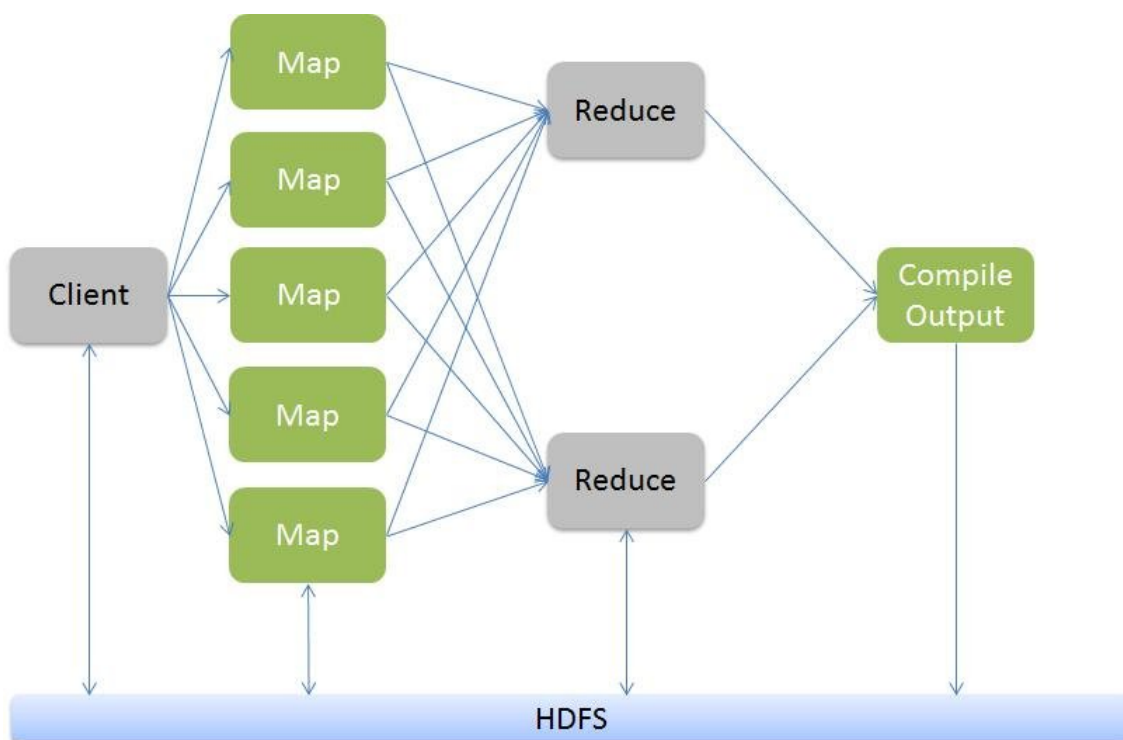
MapReduce là cơ chế xử lý song song giữa các node trong cluster dựa trên các dữ liệu đầu vào lấy từ file system, trong trường hợp Hadoop chính là HDFS. Các khái niệm chính trong

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

MapReduce bao gồm:

- **Mapper:** thực hiện map task, dữ liệu đầu vào sau quá trình xử lý trung gian sẽ đưa ra các cặp giá trị theo dạng key/value trung gian. Kết quả của quá trình này sẽ được sắp xếp theo key sử dụng ở bước sau – Reduce.
- **Reducer:** thực hiện reduce task, dữ liệu đầu vào của Map task sẽ được gom lại theo nhóm theo key. Kết quả của quá trình này là một tập dữ liệu gồm key/value collection và được ghi vào file hay lưu vào database tùy vào nhu cầu của lập trình viên.

Khi bắt đầu một MapReduce job, trước tiên Hadoop sẽ tiến hành cắt dữ liệu đầu vào thành từng gói nhỏ và phân phối vào các node được chỉ định thực hiện map task. Hình 4.14 thể hiện quy trình của một MapReduce job trong Hadoop.



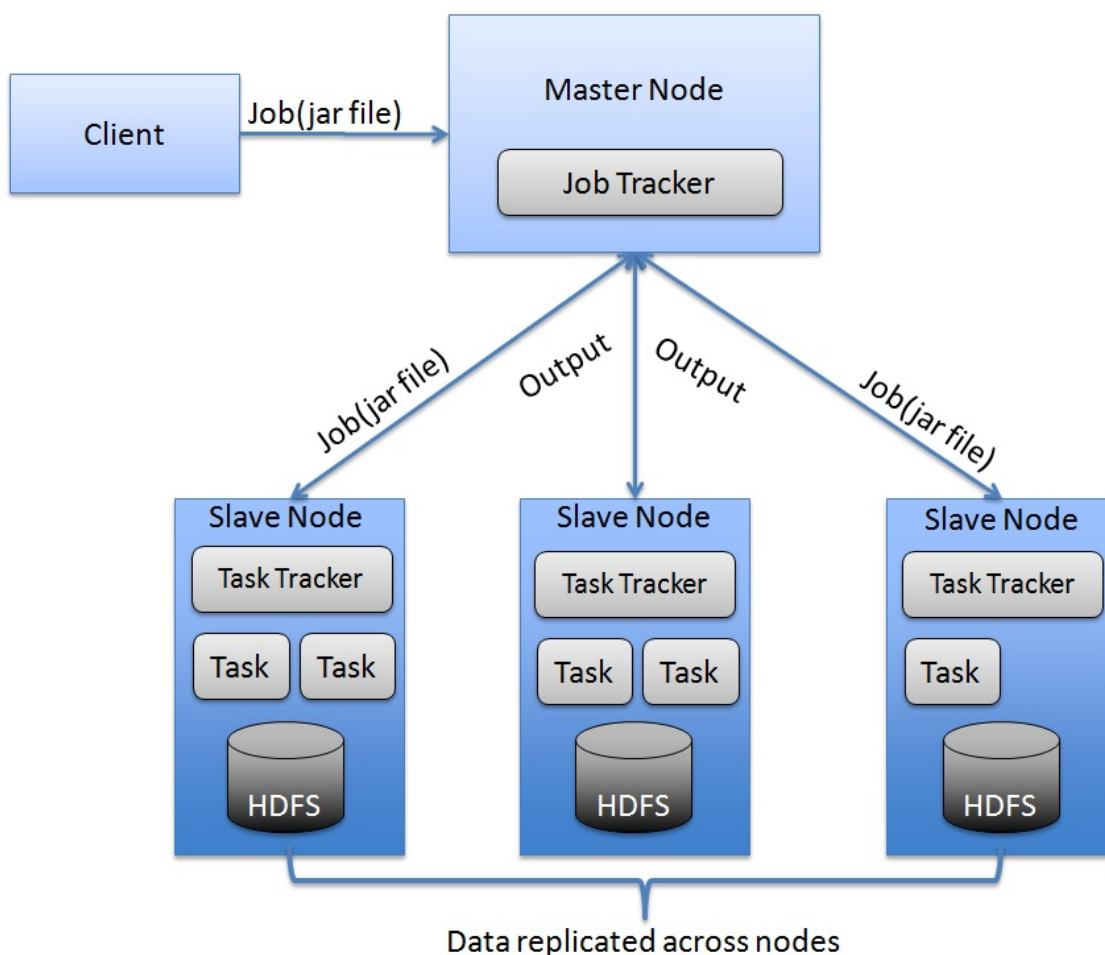
Hình 4.14: Quy trình thực thi MapReduce

Tương tự như kiến trúc của HDFS, MapReduce cũng bao gồm hai thành phần chính là JobTracker và TaskTracker (hình 4.15):

- **JobTracker** (master): đóng vai trò phân phối, điều hành các công việc cho các task-tracker trong cluster, theo dõi tiến độ thực hiện và thực thi lại các task bị fail. Thông thường trong một Hadoop cluster chỉ có một JobTracker.

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

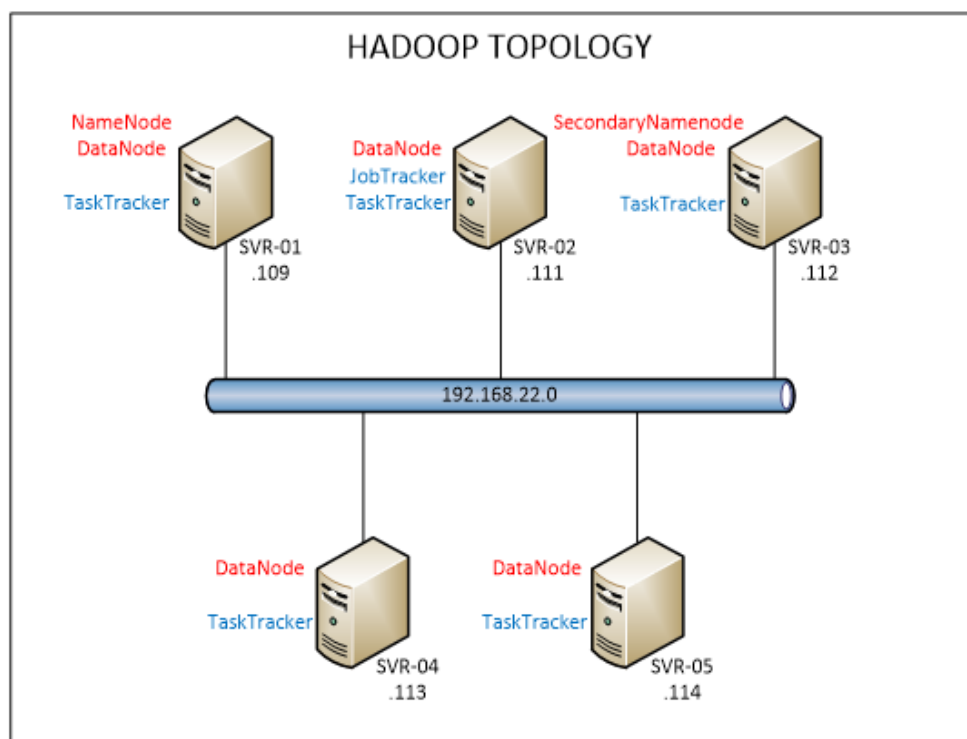
- **TaskTracker** (slaves): là các máy thực hiện công việc được giao bởi JobTracker, một TaskTracker có thể thực hiện map task, reduce task hoặc thực hiện cả hai. Thông thường các TaskTracker trong cluster cũng chính là các namenode của HDFS.



Hình 4.15: Kiến trúc Hadoop và các thành phần liên quan

4.2.4 Xây dựng Hadoop Cluster

Lượng dữ liệu từ Google Books n-grams là rất lớn, khoảng 200 Terabyte. Do đó để có thể lưu trữ toàn bộ cũng như xử lý nhanh chóng các dữ liệu từ Google Books, chúng tôi nhận thấy sử dụng Hadoop là phù hợp và cho độ hiệu quả cao. Từ các server ảo do nhà trường cung cấp cùng với sự hỗ trợ của Giảng viên hướng dẫn và các bạn trong khoa Mạng máy tính và truyền thông, chúng tôi đã xây dựng một Hadoop cluster nhỏ với 5 máy trong mạng nội bộ của trường Đại học Công nghệ Thông tin. Mô hình thiết kế của cluster này được mô tả ở hình 4.16.



Hình 4.16: Hadoop cluster được xây dựng trong mạng nội bộ

4.2.5 Thu thập dữ liệu từ Google Books n-grams bằng Hadoop

Để thu thập các n-grams cần thiết để bổ sung cho cơ sở dữ liệu của hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết, chúng tôi đã xây dựng thư viện MapReduce bằng ngôn ngữ lập trình Java, thư viện Hadoop phiên bản 1.2 và chạy trên cluster đã setup. Dựa vào cách biểu diễn dữ liệu trong các file của Google Book n-grams được trình bày trong bảng 4.2, ứng dụng thu thập n-grams của chúng tôi gồm hai class chính như trong bảng 4.3.

Nguyên tắc	ngram TAB year TAB match_count TAB volume_count NEWLINE
Ví dụ	circumvallate 1978 335 91
	circumvallate 1979 261 91

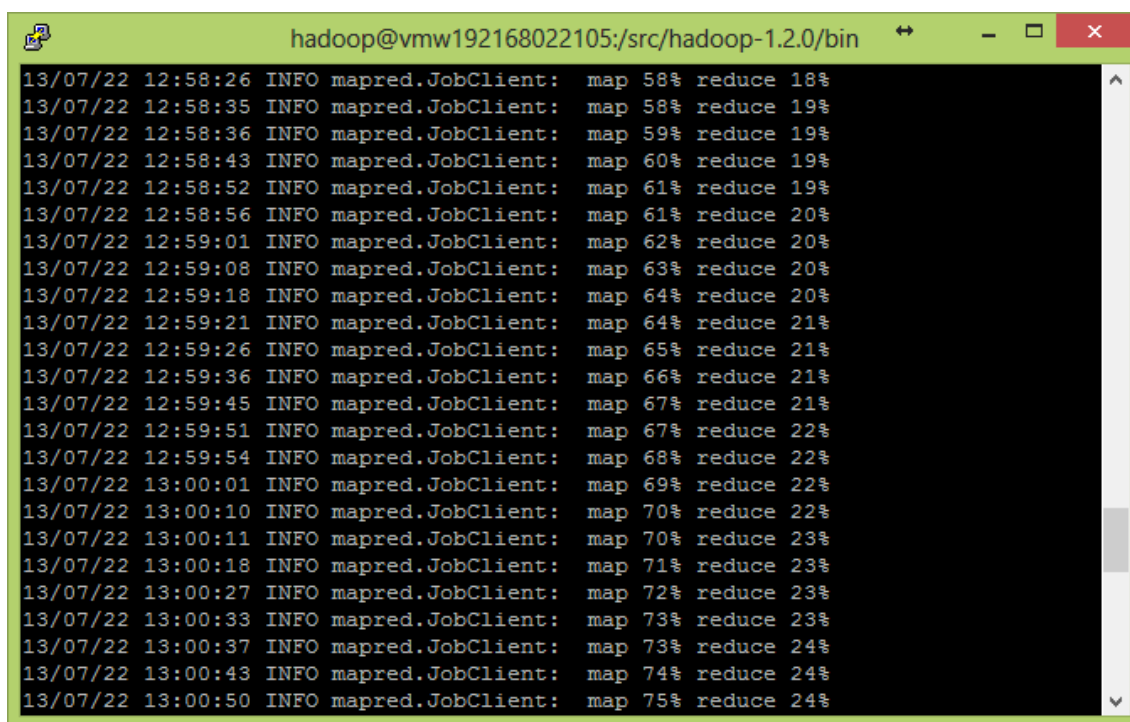
Bảng 4.2: Biểu diễn dữ liệu trong Google Book n-grams file

Trong quá trình thực thi, Hadoop cung cấp cho lập trình viên nhiều công cụ để theo dõi tiến độ thực hiện. Các hình ảnh từ 4.17 đến 4.20 thể hiện các thông tin mà Hadoop cung cấp trong suốt quá trình thực thi một job.

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

Tên class	Input	Output
GoogleBookMapper	Các dòng dữ liệu do Hadoop cắt nhỏ ra từ file đầu vào	Các cặp giá trị key/value, trong đó key là n-grams và value là số lần xuất hiện của n-grams đó
GoogleBookReducer	Dữ liệu output từ kết quả của Mapper	Các cặp giá trị key/value với key vẫn là n-grams nhưng value là tổng số lần xuất hiện trong tất cả các tài liệu được thông kê bởi GoogleBook của n-grams đó.

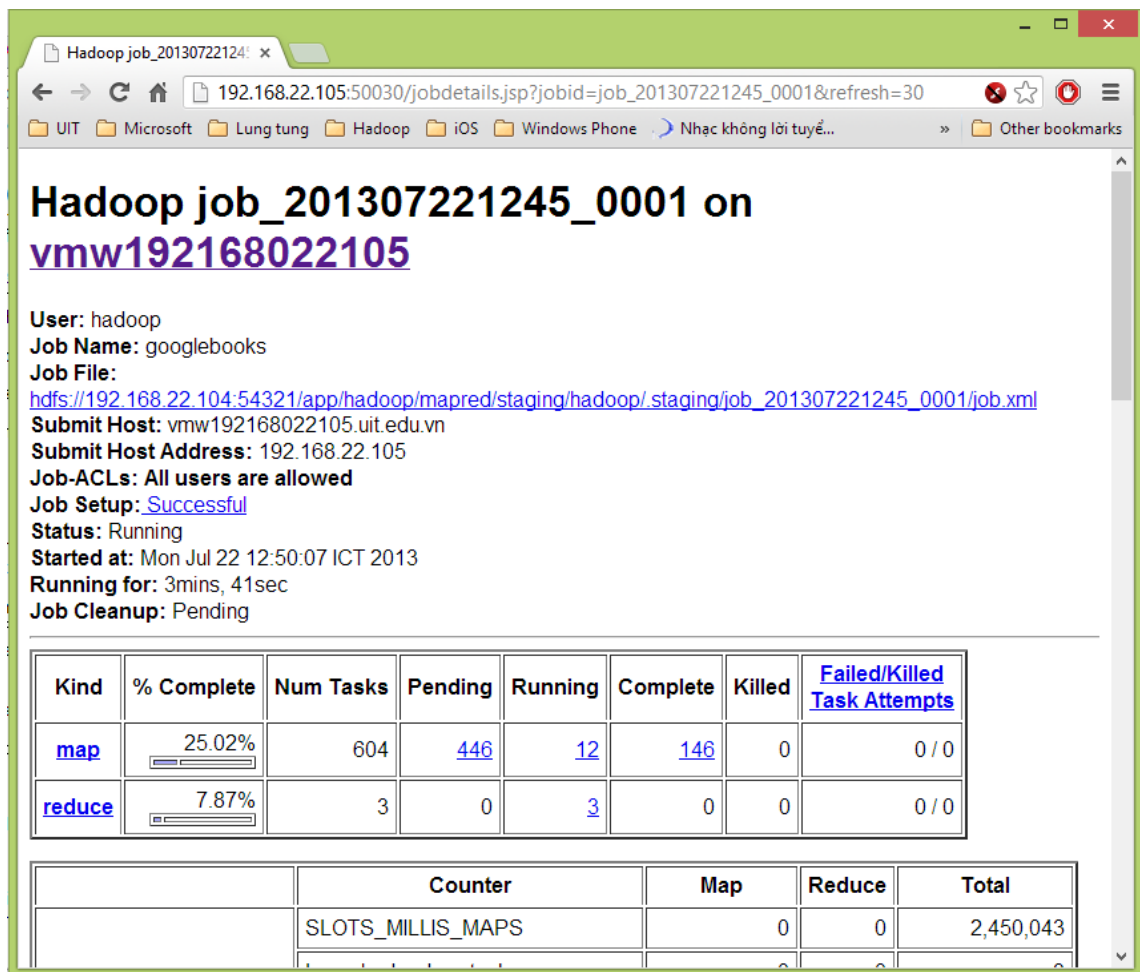
Bảng 4.3: Mapper và Reducer để thu thập n-grams



```
hadoop@vmw192168022105:/src/hadoop-1.2.0/bin
13/07/22 12:58:26 INFO mapred.JobClient: map 58% reduce 18%
13/07/22 12:58:35 INFO mapred.JobClient: map 58% reduce 19%
13/07/22 12:58:36 INFO mapred.JobClient: map 59% reduce 19%
13/07/22 12:58:43 INFO mapred.JobClient: map 60% reduce 19%
13/07/22 12:58:52 INFO mapred.JobClient: map 61% reduce 19%
13/07/22 12:58:56 INFO mapred.JobClient: map 61% reduce 20%
13/07/22 12:59:01 INFO mapred.JobClient: map 62% reduce 20%
13/07/22 12:59:08 INFO mapred.JobClient: map 63% reduce 20%
13/07/22 12:59:18 INFO mapred.JobClient: map 64% reduce 20%
13/07/22 12:59:21 INFO mapred.JobClient: map 64% reduce 21%
13/07/22 12:59:26 INFO mapred.JobClient: map 65% reduce 21%
13/07/22 12:59:36 INFO mapred.JobClient: map 66% reduce 21%
13/07/22 12:59:45 INFO mapred.JobClient: map 67% reduce 21%
13/07/22 12:59:51 INFO mapred.JobClient: map 67% reduce 22%
13/07/22 12:59:54 INFO mapred.JobClient: map 68% reduce 22%
13/07/22 13:00:01 INFO mapred.JobClient: map 69% reduce 22%
13/07/22 13:00:10 INFO mapred.JobClient: map 70% reduce 22%
13/07/22 13:00:11 INFO mapred.JobClient: map 70% reduce 23%
13/07/22 13:00:18 INFO mapred.JobClient: map 71% reduce 23%
13/07/22 13:00:27 INFO mapred.JobClient: map 72% reduce 23%
13/07/22 13:00:33 INFO mapred.JobClient: map 73% reduce 23%
13/07/22 13:00:37 INFO mapred.JobClient: map 73% reduce 24%
13/07/22 13:00:43 INFO mapred.JobClient: map 74% reduce 24%
13/07/22 13:00:50 INFO mapred.JobClient: map 75% reduce 24%
```

Hình 4.17: Quản lý tiến độ thực hiện MapReduce trên giao diện console

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



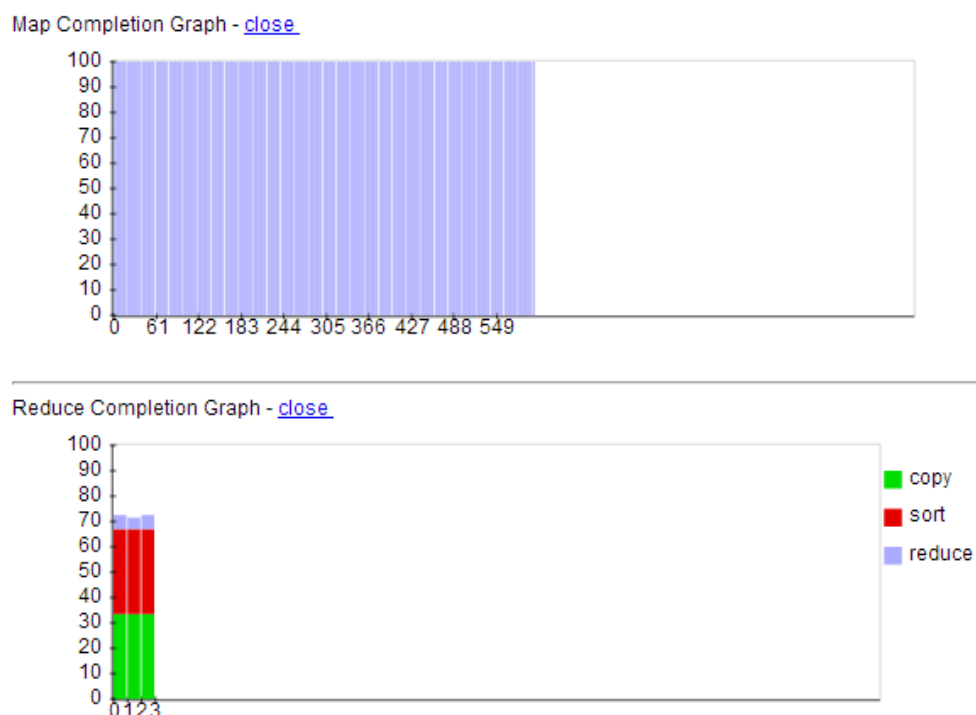
Hình 4.18: Quản lý tiến độ thực hiện MapReduce trên giao diện Web

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	3,207,374
	Launched reduce tasks	0	0	3
	Launched map tasks	0	0	204
	Data-local map tasks	0	0	204
File Input Format Counters	Bytes Read	0	0	12,962,368,347
FileSystemCounters	FILE_BYTES_READ	45,361,883	0	45,361,883
	HDFS_BYTES_READ	12,962,038,573	0	12,962,038,573
	FILE_BYTES_WRITTEN	97,373,401	162,285	97,535,686
Map-Reduce Framework	Reduce input groups	0	0	0
	Map output materialized bytes	0	0	43,420,654
	Combine output records	0	0	2,601,978
	Map input records	0	0	510,468,344
	Reduce shuffle bytes	0	0	42,026,812
	Physical memory (bytes) snapshot	0	0	71,169,892,352
	Reduce output records	0	0	0
	Spilled Records	0	0	3,726,266
	Map output bytes	0	0	1,957,731,149
	Total committed heap usage (bytes)	0	0	60,713,992,192
	CPU time spent (ms)	0	0	2,907,270
	Virtual memory (bytes) snapshot	0	0	216,860,033,024
	SPLIT_RAW_BYTES	26,578	0	26,578
	Map output records	0	0	94,626,437
	Combine input records	0	0	95,005,988
	Reduce input records	0	0	0

Hình 4.19: Thông kê dữ liệu của MapReduce job đang chạy

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



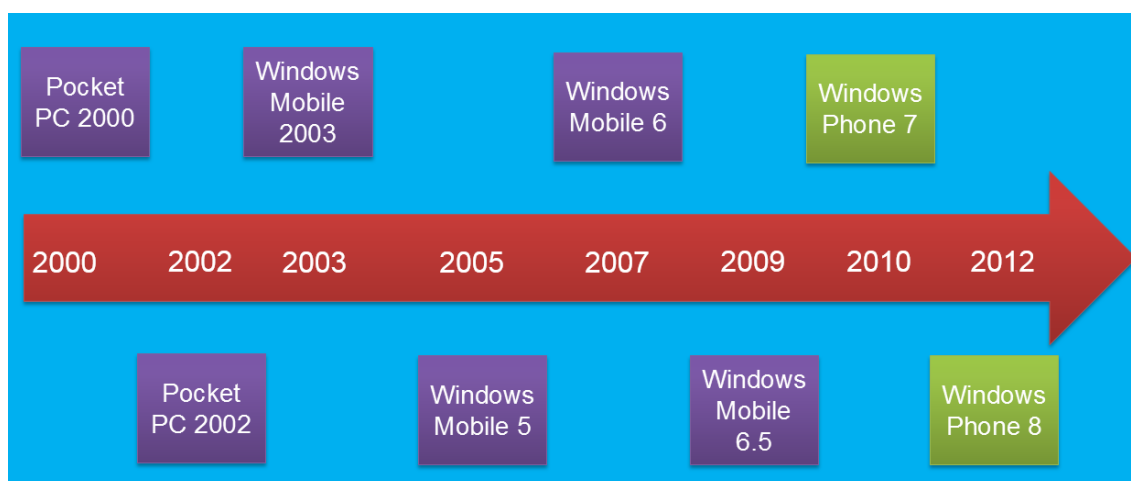
Hình 4.20: Thống kê tiến độ thực hiện bằng đồ thị

4.3 Ứng dụng giải bài tập trắc nghiệm tiếng Anh trên Windows Phone

4.3.1 Giới thiệu hệ điều hành Windows Phone

Được giới thiệu lần đầu tiên vào năm 2010, hệ điều hành Windows Phone là hệ điều hành di động mới nhất của Microsoft khi bước vào thị trường di động đang cạnh tranh khốc liệt. Windows Phone không phải là bản nâng cấp từ phiên bản hệ điều hành Windows Mobile trước đó. Microsoft đã tạo ra một phiên bản hệ điều hành hoàn toàn mới, thay đổi hoàn toàn các nguyên tắc xây dựng ứng dụng. Để tạo nên sức mạnh cho chiếc điện thoại, Microsoft bắt đầu với các nền tảng quen thuộc như Windows CE (sau này là Windows NT), .NET Framework, đồng bộ hóa bằng Zune, sử dụng công nghệ Silverlight và XNA để xây dựng các ứng dụng, đồng thời cũng thêm toàn bộ các API mới phục vụ cho việc tương tác với phần cứng di động, cảm biến và phần mềm. Microsoft cũng tạo một toolbox tích hợp trong Visual Studio, Expression Blend và XNA Game Studio để hỗ trợ cho các lập trình viên [14].

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



Hình 4.21: Lịch sử phát triển các hệ điều hành di động của Microsoft

Khi Microsoft ban đầu tiết lộ về Windows Phone 7 nhiều người chỉ suy nghĩ rằng nó chỉ là một hệ điều hành đơn giản để cố bắt kịp với iOS và Android mà Apple và Google cung cấp. Nhưng Microsoft có một kế hoạch khác hẳn cho riêng mình. Một hệ điều hành mới cho điện thoại được xuất phát từ những tài nguyên đang có và phải khác hẳn với những nhà cung cấp hệ điều hành di động trước đó chủ yếu là Apple, Reach in Motion (RIM nay là BlackBerry) và Google. Thay vì bắt chước kiểu hiển thị các biểu tượng (icon) trên màn hình, Microsoft nghĩ ra một cách khác biệt hoàn toàn. Việc thiết kế ứng dụng và hệ điều hành được định nghĩa trong một ngôn ngữ thiết kế mới là Microsoft Design Language (tên gọi đầu tiên là Metro Design Language). Ngôn ngữ thiết kế định nghĩa một bộ các hướng dẫn và phong cách cho việc tạo ứng dụng Windows Phone. Màn hình chính thiết kế theo kiểu Metro thì tương tự như cách thiết kế của các điện thoại thông minh khác, nó cũng là một danh sách các biểu tượng. Nhưng thay vì chia các biểu tượng vào từng trang, Windows Phone để người dùng di chuyển qua các biểu tượng. Windows Phone cũng tạo sự khác biệt so với các hệ điều hành di động khác ở chỗ mỗi biểu tượng có thể bao gồm một số thông tin về ứng dụng. Những biểu tượng như vậy được gọi là Live Tiles (hình 4.22).

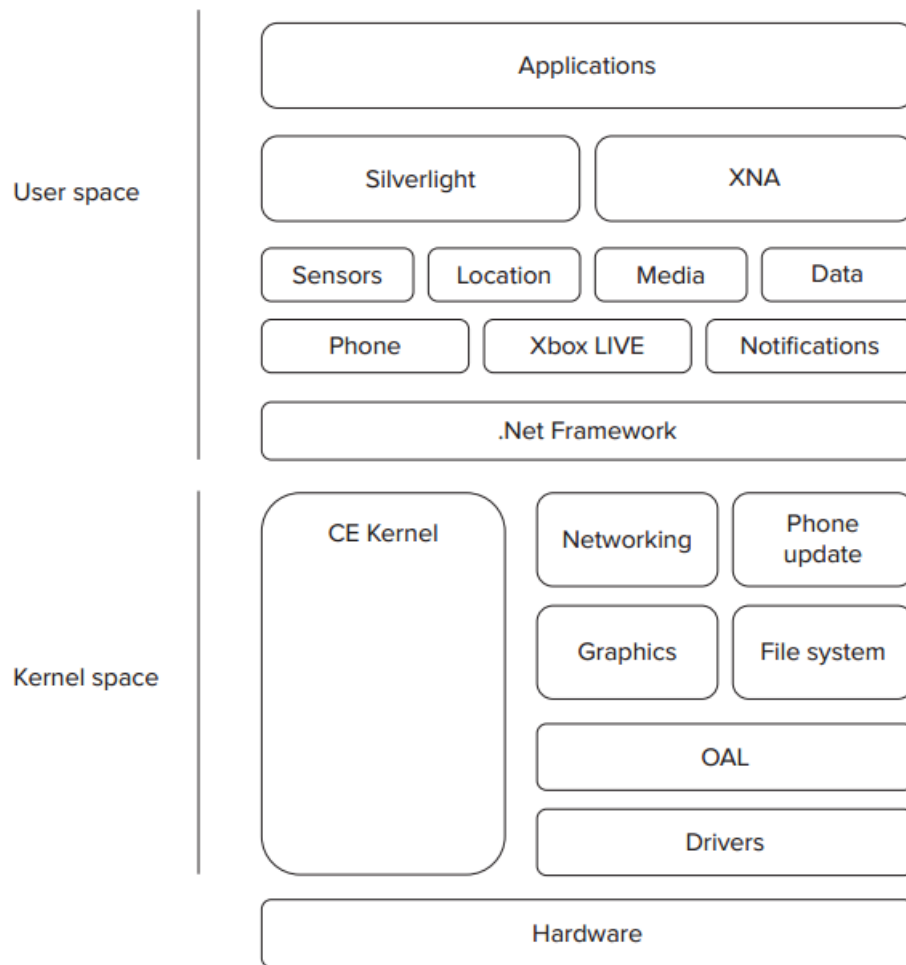


Hình 4.22: Giao diện Home của HDH Windows Phone

4.3.2 Kiến trúc Windows Phone

Phiên bản đầu tiên của hệ điều hành Windows Phone là Windows Phone 7 được dựa trên một biến thể của Microsoft Embedded OS, Windows CE 6 trong khi Windows Mobile 6.x thì dựa trên Windows CE 5. Thông thường, Windows CE cung cấp một nhân 32-bit mà được thiết kế cho các thiết bị nhúng, và một bộ các dịch vụ hệ thống như quản lý vùng nhớ, quản lý kết nối và mạng, I/O, và đồ họa. Nói cách khác, hệ điều hành Windows Phone được xây dựng nhân CE kèm theo những dịch vụ hệ thống cụ thể và một framework ứng dụng cho điện thoại di động. Kiến trúc hệ điều hành WP7 gồm 3 lớp: user space, kernel space, hardware. Toàn bộ .NET Framework trong kiến trúc nằm trong user space. Phần OS kernel, drivers, và các dịch vụ hệ thống được thực thi trong kernel space (hình 4.23).

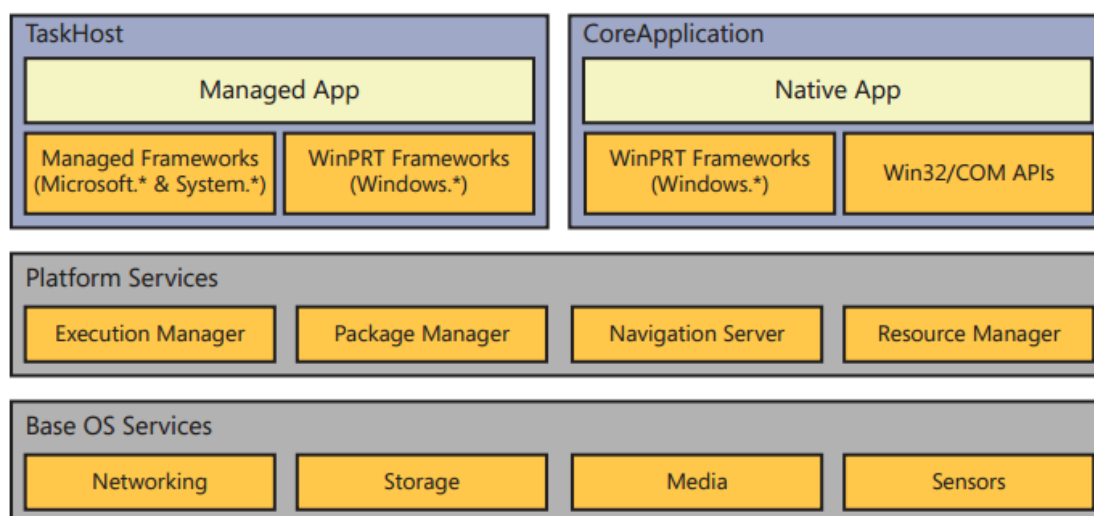
Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



Hình 4.23: Kiến trúc hệ điều hành Windows Phone 7

Nếu như Windows Phone 7 dựa trên Windows CE thì Windows Phone 8 là một thế hệ mới mà được phát triển dựa trên Windows NT, do đó cho phép lập trình viên có thể viết được các native application bằng ngôn ngữ C++ (hình 4.24).

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



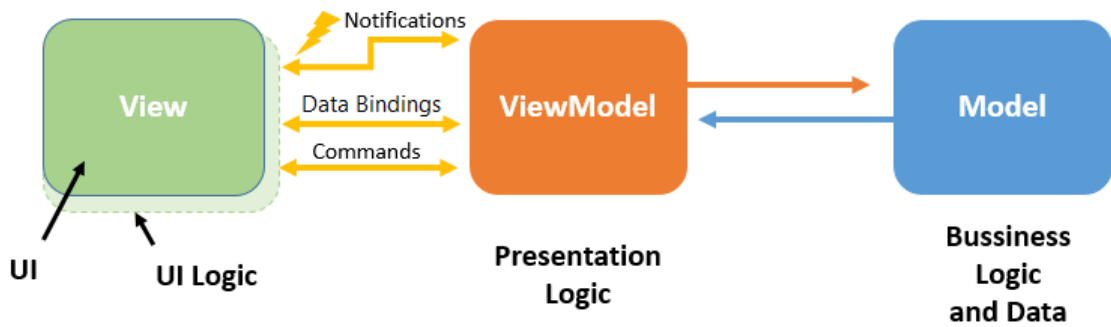
Hình 4.24: Kiến trúc hệ điều hành Windows Phone 8

4.3.3 Mô hình ba lớp Model-View-View Model (MVVM)

Ứng dụng Windows Phone được xây dựng bằng công nghệ Silverlight của Microsoft. Trong Silverlight, người lập trình xây dựng giao diện người dùng (UI) bằng ngôn ngữ XAML có cú pháp giống với ngôn ngữ đặc tả XML. Việc sử dụng một ngôn ngữ độc lập để lập trình giao diện chính là tiền đề để phát triển các dạng mô hình ba lớp cho lập trình Windows Phone cũng như các công nghệ tương tự như WPF hay Windows 8.

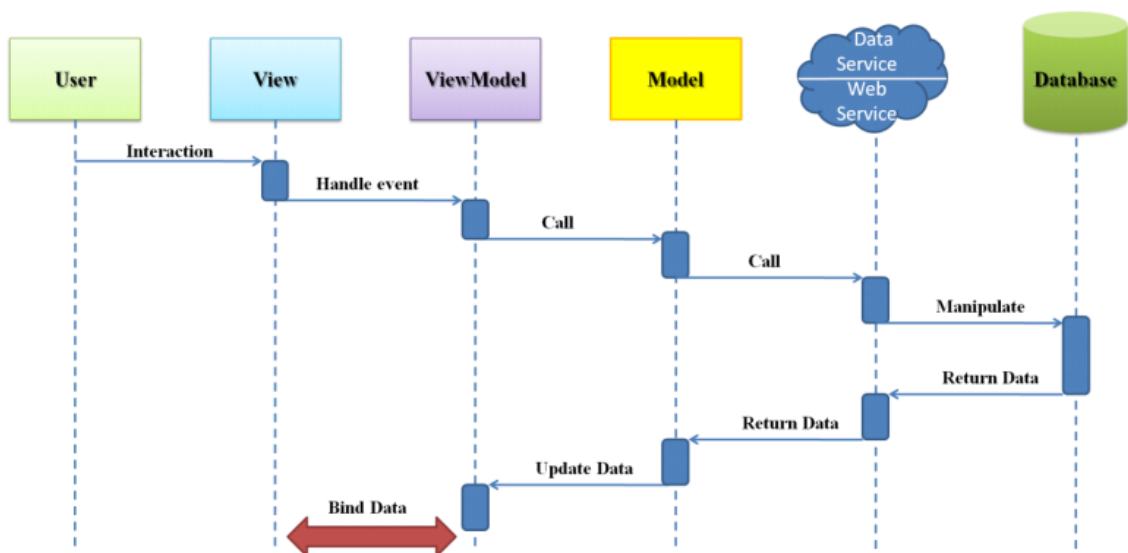
Phát triển lên từ mô hình MVP (Model-View-Presenter), năm 2004 Martin Fowler đưa ra khái niệm mô hình ba lớp Model-View-View Model (MVVM) áp dụng trong nền tảng lập trình ứng dụng có giao diện người dùng độc lập, hỗ trợ lập trình hướng sự kiện như WPF, Silverlight. MVVM tạo tách biệt giữa việc thiết kế giao diện và lập trình xử lý. Hình 4.25 biểu diễn các thành phần cấu thành nên kiến trúc MVVM và mối quan hệ cũng như sự tương tác giữa các thành phần.

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



Hình 4.25: Kiến trúc MVVM

- **View:** Tương tự như trong mô hình MVC, View là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. View không hề biết sự tồn tại của Model, và ngược lại Model cũng không biết gì về View. Một điểm khác biệt so với các ứng dụng truyền thống là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng binding, command
- **Model:** Cũng tương tự như trong mô hình MVC. Model là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.
- **View Model:** Lớp trung gian giữa View và Model. ViewModel có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.



Hình 4.26: Quy trình hoạt động của mô hình MVVM

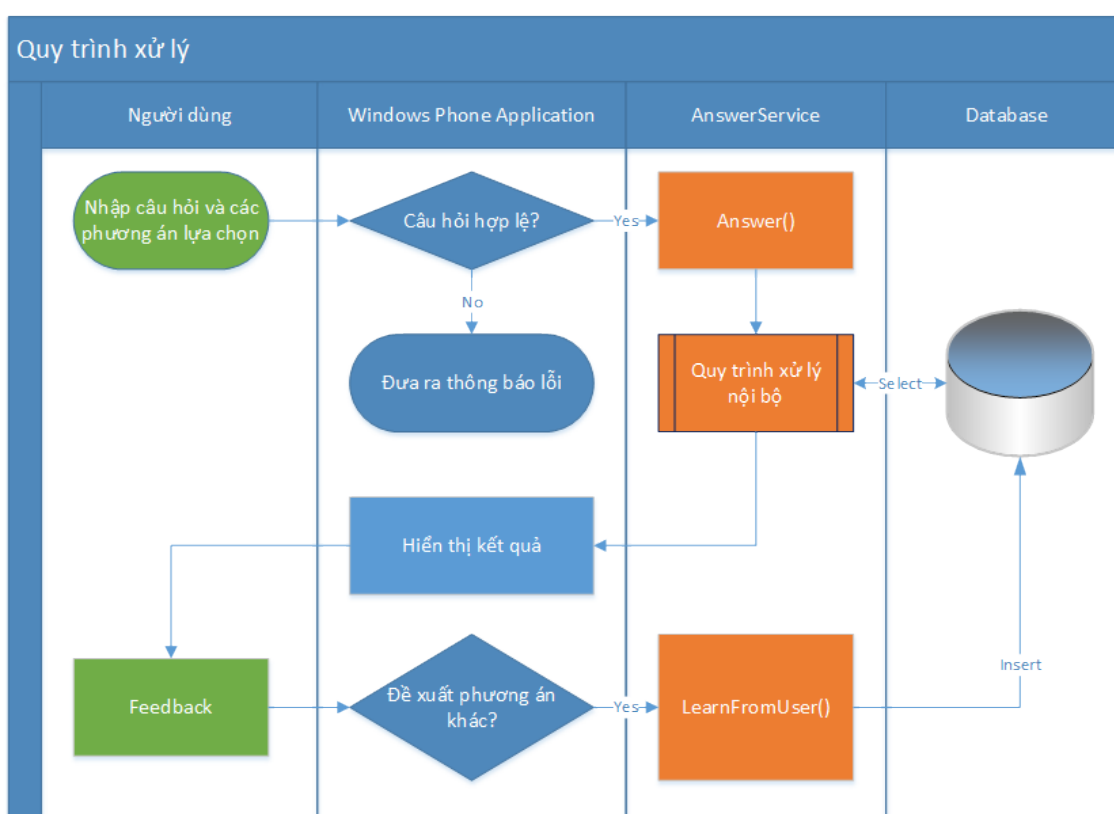
Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

4.3.4 Xây dựng ứng dụng

Ứng dụng trả lời câu hỏi trắc nghiệm tiếng Anh điền khuyết cho người dùng Windows Phone sẽ cung cấp các chức năng cơ bản, bao gồm:

- Nhận câu hỏi và các đáp án từ người dùng
- Gọi AnswerService để trả lời cho câu hỏi đầu vào
- Cho phép người dùng gợi ý câu trả lời đúng cho câu hỏi trong trường hợp câu trả lời do hệ thống đưa ra không chính xác.

Hình 4.27 mô tả quy trình tương tác giữa người dùng, ứng dụng Windows Phone, Answer service và cơ sở dữ liệu.



Hình 4.27: Quy trình xử lý của hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh

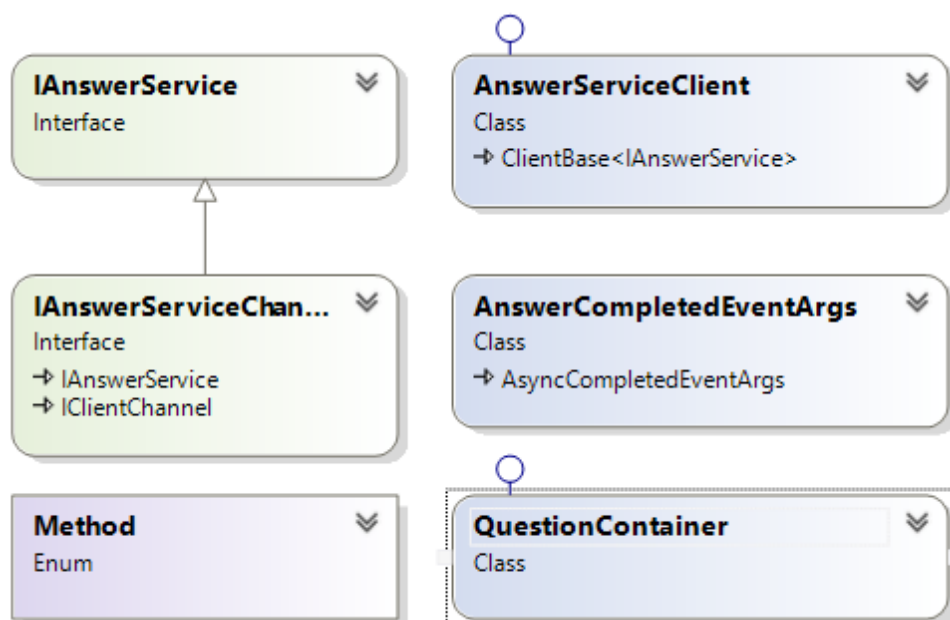
4.3.4.1 Model

Trong mô hình MVVM, Model đóng vai trò là trung tâm dữ liệu, tuy nhiên vì ứng dụng Windows Phone không được phép truy xuất trực tiếp vào dữ liệu mà chỉ có thể tương tác thông qua việc gọi WCF Service nên ta sẽ xem phần Model của ứng dụng là Answer Service.

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

Với công cụ lập trình Visual Studio, việc kết nối và sử dụng WCF Service được thực hiện rất đơn giản bằng thao tác “Add Service References” và trỏ đường dẫn đến service ta cần dùng. Visual Studio sẽ tự động sinh ra các class như trong hình 4.28. Để tương tác với service, ta chỉ cần khởi tạo một thực thể thuộc lớp **AnswerServiceClient** và gọi các chức năng mà service cung cấp tương tự như việc gọi đến một phương thức của đối tượng (hình 4.29).

Để đảm bảo về trải nghiệm người dùng (User Experience – UX), ta cần gọi các phương thức của service bằng phương pháp bất đồng bộ (asynchronous). Nghĩa là nếu ta gọi một hành động nào đó trên service, sau khi gọi xong, tài nguyên của máy sẽ được trả lại cho UI thread. Kết quả trả về sẽ được truy xuất thông qua một hàm call back với một tham số là đối tượng thuộc lớp **AnswerCompletedEventArgs**. Từ đối tượng này ta có thể lấy ra được kết quả trả về từ service.



Hình 4.28: Lược đồ UML các lớp trong Model

```
AnswerServiceClient client = new AnswerServiceClient();

// Use the 'client' variable to call operations on the service.

// Always close the client.
client.Close();
```

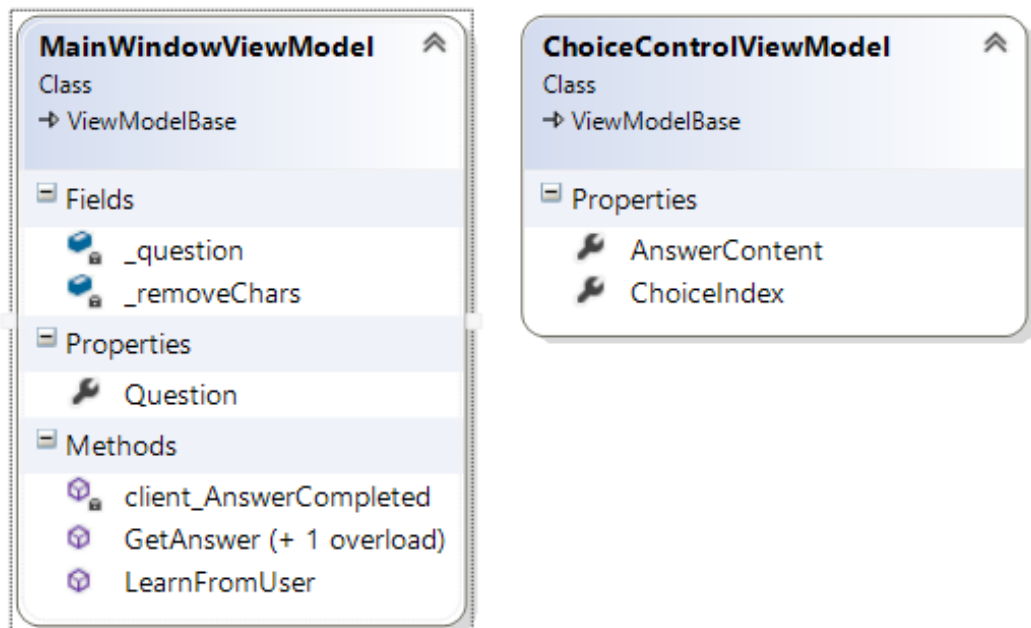
Hình 4.29: Gọi AnswerService

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

4.3.4.2 View Model

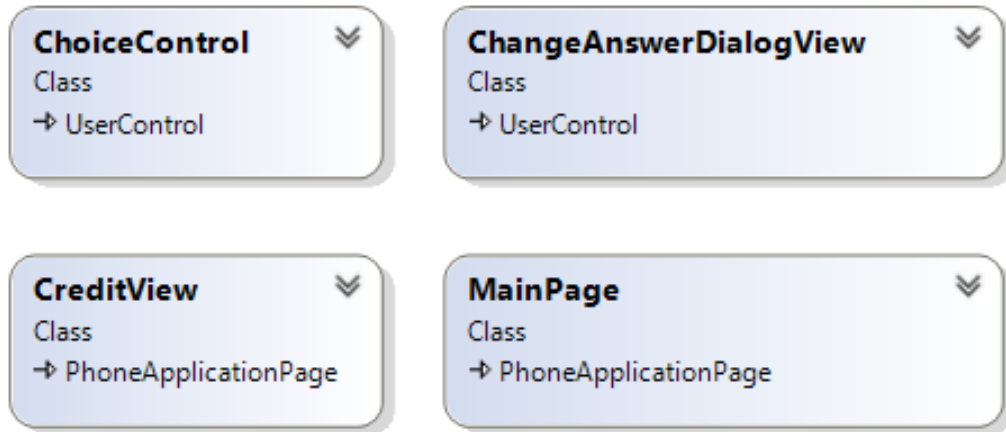
Ứng dụng trả lời câu hỏi trắc nghiệm trên Windows Phone có hai view model (hình 4.30). Trong đó,

- **MainWindowViewModel:** view model cho màn hình chính của ứng dụng, làm trung gian giao tiếp giữa view và web service.
- **ChoiceControlViewModel:** để hiển thị các đáp án có thể có của câu hỏi, chúng tôi xây dựng một custom control mới gọi là ChoiceControlView gồm 1 text block binding (kết nối) trực tiếp vào property ChoiceIndex của view model và 1 text box cho phép người dùng nhập vào phương án được binding vào property AnswerContent của view model.



Hình 4.30: Các class trong View Model layer

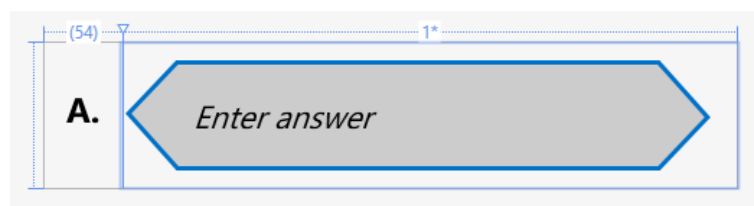
4.3.4.3 View



Hình 4.31: Các class trong View layer

Giao diện của ứng dụng trả lời câu hỏi trắc nghiệm tiếng Anh có hai trang chính:

- **MainPage**: giao diện tương tác giữa người dùng và ứng dụng
- **CreditView**: thông tin về sinh viên thực hiện đề án, giảng viên hướng dẫn.
- **ChoiceControl**: hiển thị một phương án lựa chọn cho câu hỏi. Nếu kết quả trả về là phương án đúng sẽ hiển thị các animation thay đổi màu nền để người dùng chú ý (hình 4.32).
- **ChangeAnswerDialogView**: Dùng để cho phép người dùng lựa chọn lại đáp án khác cho câu hỏi trong trường hợp đáp án hệ thống đưa ra không chính xác. Tuy nhiên Windows Phone không hỗ trợ kiểu Dialog có nhiều hơn 2 button, do đó ta cần xây dựng custom control mới giả lập như một Message Dialog nhưng có nhiều button thể hiện các đáp án của câu hỏi.

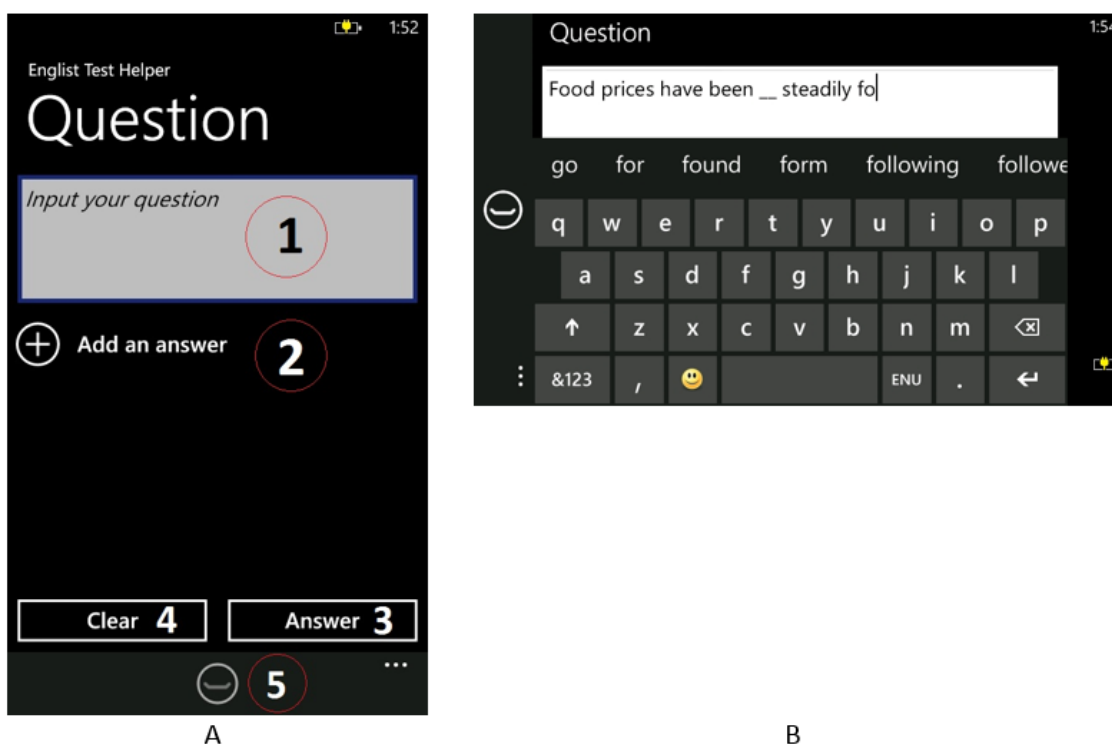


Hình 4.32: ChoiceControl

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

4.3.4.4 Screenshots

Hình 4.33 là giao diện chính của ứng dụng. Ở ô nhập câu hỏi, khi muốn nhập vào khoảng trống, nhấn nút tròn (số 5) ở dưới cùng màn hình. Trong quá trình nhập câu hỏi, Windows Phone có thể đưa ra các gợi ý về từ vựng cho người dùng (hình 4.33b)



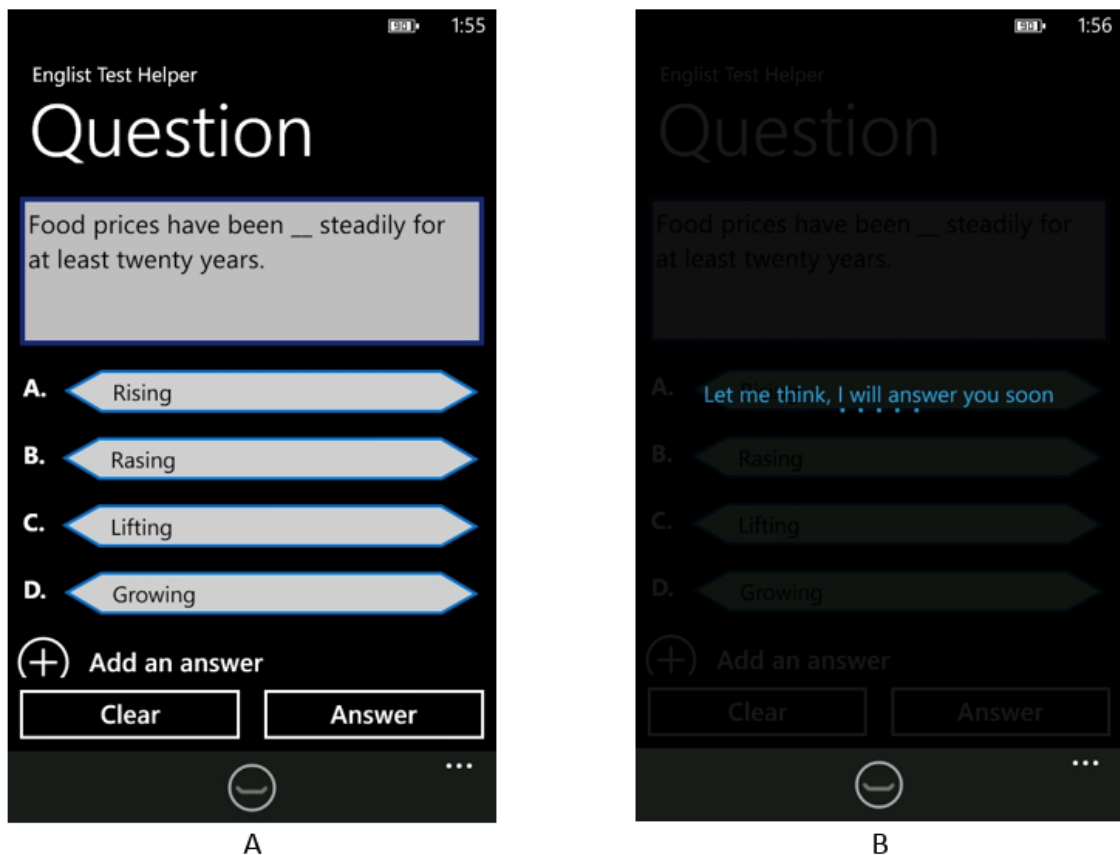
Hình 4.33: A. Giao diện portrait. B. Giao diện landscape

- Chú thích:

1. Khung nhập câu hỏi
2. Thêm một phương án trả lời cho câu hỏi
3. Yêu cầu hệ thống đưa ra câu trả lời cho câu hỏi
4. Xóa các text box, đưa màn hình về giao diện ban đầu
5. Thêm khoảng trống vào khung nhập câu hỏi

Sau khi nhập xong câu hỏi và các phương án trả lời (hình 4.34a), nhấn vào button Answer, chương trình sẽ tiến hành xử lý đầu vào và gọi AnswerService để trả về câu hỏi cho người dùng (hình 4.34b).

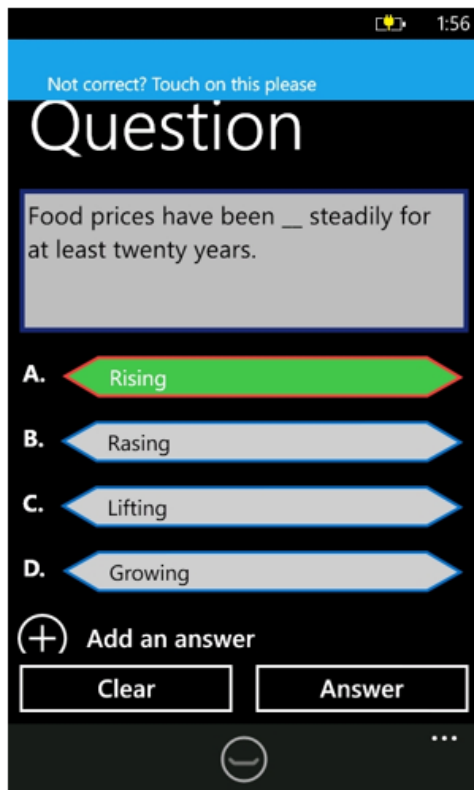
Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



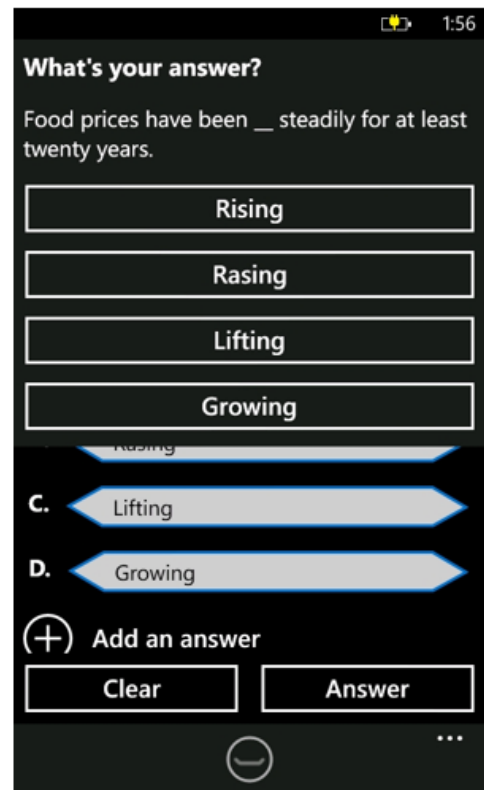
Hình 4.34: A. Sau khi nhập xong câu hỏi và các phương án trả lời. B. Ứng dụng đang chờ câu trả lời từ AnswerService

Khi nhận được response từ service, chương trình sẽ hiển thị phương án được chọn bằng một loạt các animation thay đổi màu nền của đáp án đó, đồng thời hiển thị một toast notification (hình chữ nhật màu xanh – 4.35a), nếu người dùng cho rằng đáp án hệ thống đưa ra không chính xác có thể chạm toast notification để chọn lại đáp án (hình 4.35b).

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

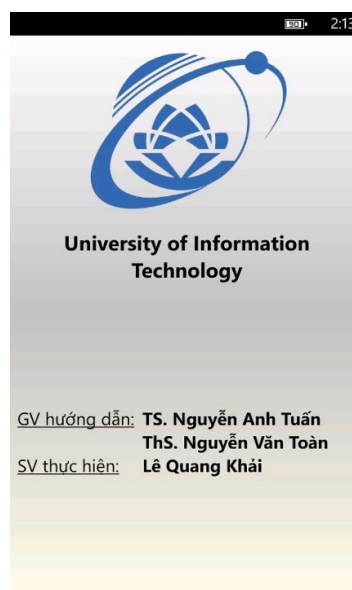


A



B

Hình 4.35: A. đưa ra đáp án đề xuất cho câu hỏi. B. Người dùng chọn lại đáp án nếu đáp án đề xuất không chính xác



Hình 4.36: Màn hình credit

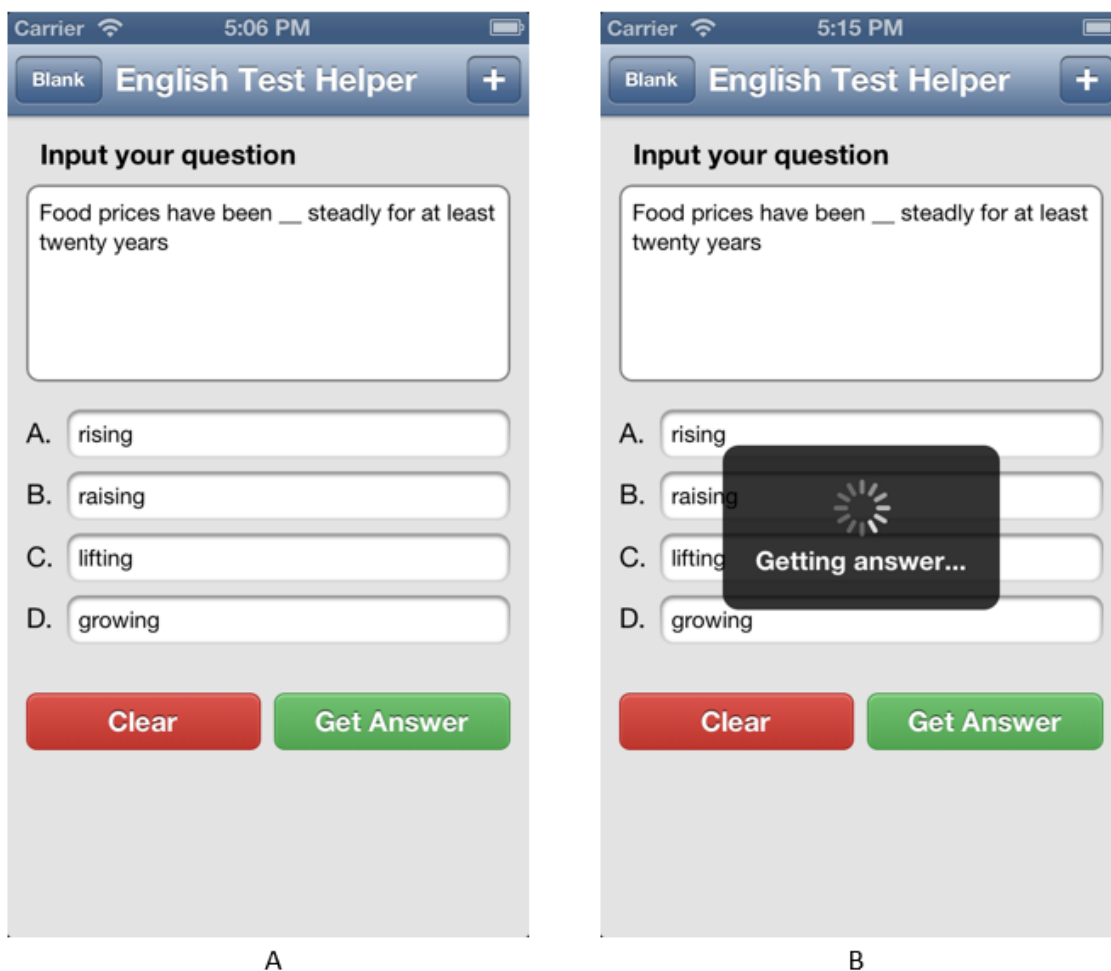
Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH

4.3.5 Mở rộng hệ thống trên các nền tảng khác

Như đã trình bày ở trên, hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh điển khuyết được xây dựng theo mô hình SOA. Để chứng minh cho tính nhanh chóng, tiện lợi của hệ thống, chúng tôi tiến hành xây dựng thêm các ứng dụng với chức năng tương tự trên các nền tảng khác.

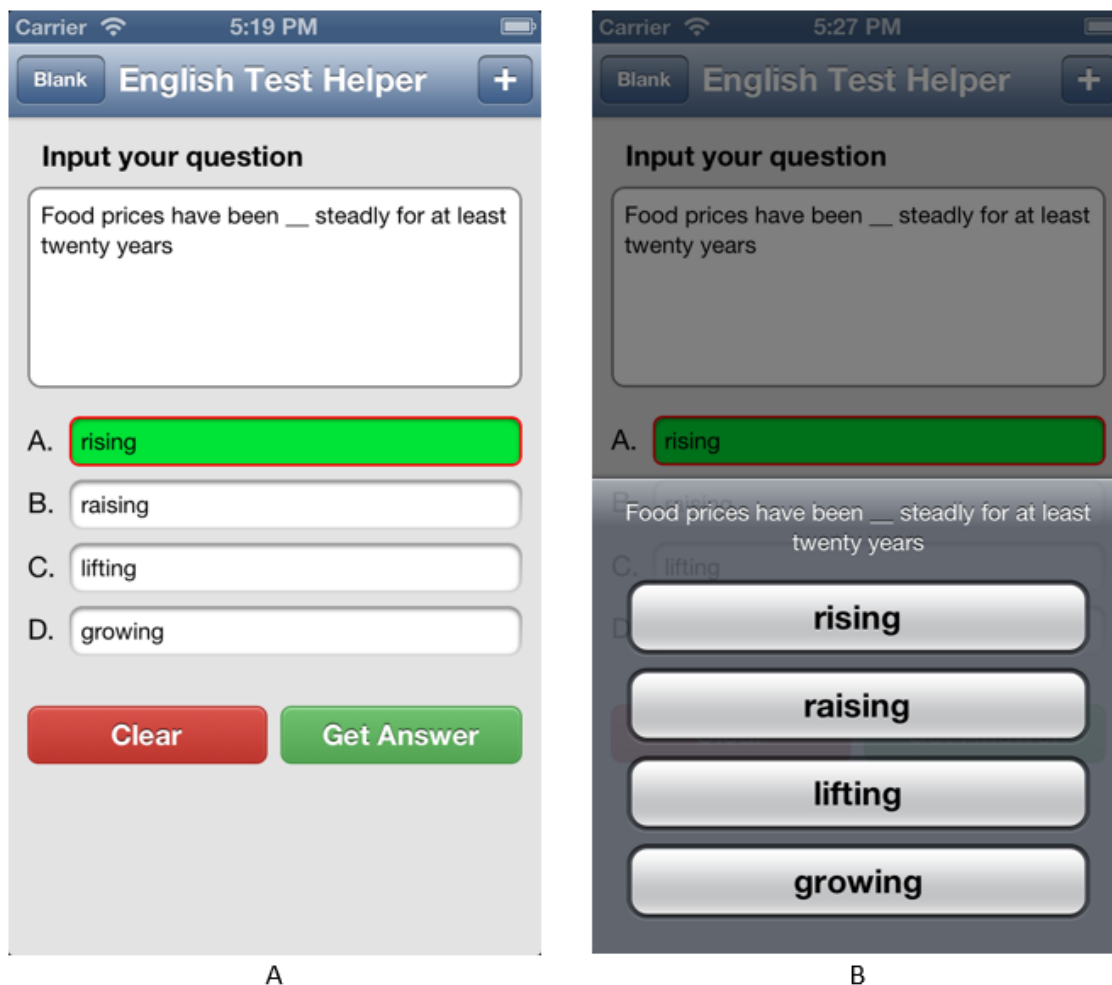
4.3.5.1 Ứng dụng trên iPhone

Ứng dụng trả lời câu hỏi trắc nghiệm tiếng Anh dạng điển khuyết trên iPhone cũng có đầy đủ các tính năng như phiên bản trên Windows Phone. Ứng dụng có thể chạy được trên các thiết bị đã update lên phiên bản iOS 6.0. Các hình ảnh bên dưới là screenshots của ứng dụng.



Hình 4.37: A. Màn hình nhập các phương án lựa chọn. B. Màn hình chờ trong lúc tìm đáp án

Chương 4. HỆ THỐNG TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TIẾNG ANH



Hình 4.38: A. Màn hình hiển thị đáp án được chọn. B. Màn hình cho phép người dùng để xuất đáp án khác

Chương 5

KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Dữ liệu khảo sát

Để khảo sát kết quả của hệ thống, chúng tôi lấy dữ liệu câu hỏi tiếng Anh điền khuyết từ các bộ đề ôn tập thi tiếng Anh, bao gồm đề thi trình độ bằng A, B, C, thi TOELF và các câu hỏi trích ra từ đề thi tiếng Anh đại học khối D các năm.

Về đề thi tiếng Anh trình độ A, B, C và TOELF, các câu hỏi và đáp án được lấy từ trang web <http://luyenthianhvan.org> bằng cách xây dựng một công cụ sử dụng XPath và Regular Expression để lọc lấy dữ liệu từ HTML source của web site. Các câu hỏi trong các bộ đề thi này đều có sẵn đáp án. Bảng 5.1 thống kê tổng số lượng câu hỏi của từng bộ đề.

Dữ liệu sau khi lấy về được tổ chức và lưu trữ bằng các file xml. Cấu trúc lưu trữ của các bộ đề thi được mô tả ở hình 5.1.

Đề thi	Số lượng câu hỏi
Bằng A	800
Bằng B	460
Bằng C	2020
TOELF	820

Bảng 5.1: Thống kê số lượng câu hỏi các bộ đề thi

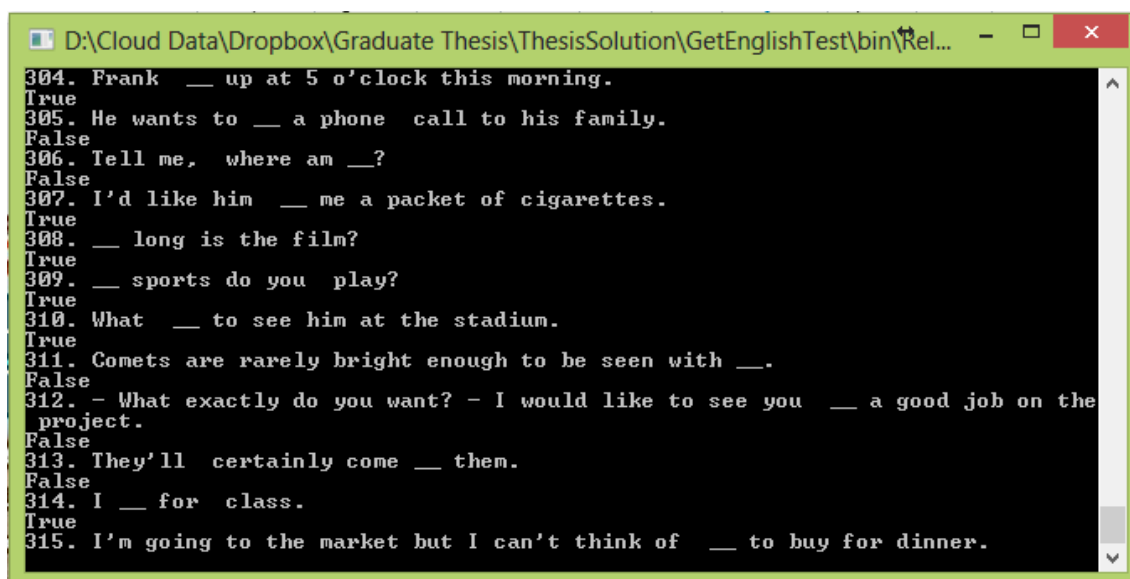
```
<?xml version="1.0" encoding="utf-8"?>
<Test Type="Level_A">
  <Questions>
    <Question>
      <Content>Who are all __ people?</Content>
      <Choices>
        <A>this</A>
        <B>those</B>
        <C>them</C>
        <D>that</D>
      </Choices>
      <Key>1</Key>
    </Question>
    <Question>
      <Content>I don't know __ people.</Content>
      <Choices>
        <A>many</A>
        <B>much</B>
        <C>a lot</C>
        <D>few</D>
      </Choices>
      <Key>0</Key>
    </Question>
    ...
  </Test>
```

Hình 5.1: Cấu trúc XML để lưu trữ các bộ đề thi

Ở bộ dữ liệu khảo sát thứ 2, chúng tôi tiến hành tìm kiếm và lọc bằng tay để chọn ra 100 câu hỏi trắc nghiệm điền khuyết từ các đề thi đại học khối D môn tiếng Anh từ năm 2009 đến năm 2012 và cũng lưu lại bằng file xml tương tự như các bộ đề thi khác.

5.2 Chương trình khảo sát tự động

Để tiến hành khảo sát từ các bộ đề thi, chúng tôi xây dựng một công cụ đọc nội dung các file xml và gọi đến AnswerService, đáp án trả về từ service sẽ được lưu lại để thống kê sau này. Công cụ khảo sát cho phép lựa chọn giữa 1 trong 3 hình thức lấy n-grams như mô tả ở phần 3.3 (hình 5.2).



Hình 5.2: Chương trình khảo sát tự động

5.3 Thực hiện khảo sát

5.3.1 Khảo sát 1 – Độ chính xác của các phương án chọn n-grams

Như đã giới thiệu ở mục 3.3, ta có 3 chiến lược lấy n-grams. Ta lần lượt tiến hành khảo sát từng phương án cho từng bộ đề kiểm tra khác nhau. Lúc thực hiện khảo sát, cơ sở dữ liệu có tổng cộng 7.352.190 bigrams và 6.534.417 trigrams. Kết quả khảo sát được trình bày trong các bảng 5.2, 5.3, 5.4, 5.5 và 5.6

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Chỉ dùng bigrams	Đáp án đúng	233	27.875
	Đáp án sai	577	72.125
Tổng cộng		800	100
Chỉ dùng trigrams	Đáp án đúng	296	37
	Đáp án sai	504	63
Tổng cộng		800	100
Kết hợp bigrams và trigrams	Đáp án đúng	317	39.625
	Đáp án sai	483	60.375
Tổng cộng		800	100

Bảng 5.2: Kết quả khảo sát trên đề ôn thi bằng A tiếng Anh

Chương 5. KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Chỉ dùng bigrams	Đáp án đúng	135	29.35
	Đáp án sai	325	70.65
Tổng cộng		460	100
Chỉ dùng trigrams	Đáp án đúng	165	35.87
	Đáp án sai	295	64.13
Tổng cộng		460	100
Kết hợp bigrams và trigrams	Đáp án đúng	181	39.35
	Đáp án sai	279	60.65
Tổng cộng		460	100

Bảng 5.3: Kết quả khảo sát trên đề ôn thi bằng B tiếng Anh

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Chỉ dùng bigrams	Đáp án đúng	614	30.40
	Đáp án sai	1406	69.60
Tổng cộng		2020	100
Chỉ dùng trigrams	Đáp án đúng	829	41.04
	Đáp án sai	1191	58.96
Tổng cộng		2020	100
Kết hợp bigrams và trigrams	Đáp án đúng	915	45.30
	Đáp án sai	1105	54.70
Tổng cộng		2020	100

Bảng 5.4: Kết quả khảo sát trên đề ôn thi bằng C tiếng Anh

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Chỉ dùng bigrams	Đáp án đúng	199	24.27
	Đáp án sai	621	75.73
Tổng cộng		820	100
Chỉ dùng trigrams	Đáp án đúng	286	34.88
	Đáp án sai	534	65.12
Tổng cộng		820	100
Kết hợp bigrams và trigrams	Đáp án đúng	278	33.90
	Đáp án sai	542	66.10
Tổng cộng		820	100

Bảng 5.5: Kết quả khảo sát trên đề ôn thi TOELF

Chương 5. KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Chỉ dùng bigrams	Đáp án đúng	29	29
	Đáp án sai	71	71
Tổng cộng		100	100
Chỉ dùng trigrams	Đáp án đúng	37	37
	Đáp án sai	63	63
Tổng cộng		100	100
Kết hợp bigrams và trigrams	Đáp án đúng	36	36
	Đáp án sai	64	64
Tổng cộng		100	100

Bảng 5.6: Kết quả khảo sát trên đề thi đại học khối D từ năm 2009 đến năm 2012

5.3.2 Khảo sát 2 - Sự phụ thuộc về độ lớn cơ sở dữ liệu với độ chính xác

5.3.2.1 Khảo sát đợt một

Trong lần khảo sát đợt 1 để đánh giá sự phụ thuộc về độ lớn cơ sở dữ liệu với độ chính xác của đáp án, chúng tôi tiến hành khảo sát với cơ sở dữ liệu chỉ mới chạy xong khoảng một nửa dữ liệu từ bộ ngữ liệu OANC. Số lượng bigrams và trigrams mỗi loại trong cơ sở dữ liệu ở vào khoảng hơn 4 triệu cho mỗi loại. Lần lượt khảo sát trên tất cả các bộ đề thi đã có. Từ kết quả của khảo sát 1, lần khảo sát này chỉ chọn phương án sử dụng kết hợp bigrams và trigrams để có độ chính xác cao nhất. Kết quả khảo sát được trình bày ở bảng 5.7

Đề thi	Tổng số câu	Số câu đúng	Tỉ lệ (%)
Bảng A	800	299	37.36
Bảng B	460	181	39.35
Bảng C	2020	858	42.46
TOELF	820	273	33.29
Đề thi đại học	100	35	35

Bảng 5.7: Kết quả khảo sát lần 1 với cơ sở dữ liệu nhỏ

5.3.2.2 Khảo sát đợt hai

Tương tự như khảo sát đợt 1, tuy nhiên trong lần khảo sát này, toàn bộ bộ ngữ liệu OANC đã được xử lý xong. Số lượng bigrams và trigrams có trong cơ sở dữ liệu lần lượt là 7.352.190 và 6.534.417. Các kết quả khảo sát được thể hiện trong nội dung bảng 5.8.

Chương 5. KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Đề thi	Tổng số câu	Số câu đúng	Tỉ lệ (%)
Bảng A	800	317	39.63
Bảng B	460	181	39.35
Bảng C	2020	915	45.30
TOELF	820	278	33.90
Đề thi đại học	100	36	36

Bảng 5.8: Kết quả khảo sát lần 2 trên cơ sở dữ liệu lớn hơn

5.3.3 Khảo sát độ chính xác của từng loại câu hỏi

Từ các đánh giá, khảo sát trên, chúng tôi phân các câu hỏi tiếng Anh thành hai nhóm chính:

- Câu hỏi liên quan đến ngữ nghĩa: là các câu hỏi có đáp án khi thế vào chỗ trống không làm sai ngữ pháp của câu nhưng có thể không phù hợp về mặt ngữ nghĩa so với ngữ cảnh của câu đang xét.
- Câu hỏi liên quan đến ngữ pháp: là các câu hỏi mà các đáp án khi thay vào sẽ có thể làm sai cấu trúc ngữ pháp của câu.

Việc phân nhóm phải được tiến hành thủ công, do đó chúng tôi sẽ chỉ khảo sát trên kết quả của 100 câu hỏi đề thi đại học (bảng 5.9) và 200 câu đầu tiên trong kết quả của 2020 câu hỏi trong bộ đề C tiếng Anh (bảng 5.10), phương án lấy kết hợp n-grams và được thực hiện trên bộ dữ liệu đầy đủ.

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Câu hỏi ngữ pháp	Đáp án đúng	17	33.33
	Đáp án sai	34	66.67
Tổng cộng		51	100
Câu hỏi ngữ nghĩa	Đáp án đúng	24	48.98
	Đáp án sai	25	51.02
Tổng cộng		49	100

Bảng 5.9: Kết quả khảo sát các loại câu hỏi trên 100 câu hỏi từ đề thi đại học

5.4 Đánh giá

Ở khảo sát đánh giá độ chính xác của từng chiến lược chọn n-grams, kết quả khảo sát cho ta thấy rất rõ ràng. Theo đó nếu chỉ lấy các bigrams, độ chính xác là thấp nhất. Với việc chỉ

Chương 5. KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Loại câu hỏi		Số lượng (câu)	Tỉ lệ (%)
Câu hỏi ngữ pháp	Đáp án đúng	34	43.59
	Đáp án sai	44	56.41
Tổng cộng		78	100
Câu hỏi ngữ nghĩa	Đáp án đúng	61	46.92
	Đáp án sai	69	53.08
Tổng cộng		130	100

Bảng 5.10: Kết quả khảo sát các loại câu hỏi trên 200 câu hỏi đầu tiên trong đề ôn thi bằng C

sử dụng các trigrams, độ chính xác có cải thiện hơn khá nhiều. Cuối cùng, với phép thử kết hợp sử dụng cả bigrams và trigrams, ta có được độ chính xác cao nhất, mặc dù có trường hợp ngoại lệ khi khảo sát trên 100 câu hỏi từ đề thi đại học nhưng không có ý nghĩa nhiều vì chỉ hơn kém nhau 1 câu đúng. Kết quả của bài khảo sát này là tiền đề để đánh giá tiếp các bài khảo sát sau.

Tiếp theo, ở bài khảo sát đánh giá độ phụ thuộc của ngữ liệu huấn luyện so với độ chính xác, ta thấy kết quả không chênh lệch nhau nhiều lắm. Cụ thể, ở 100 câu hỏi trong bộ đề thi đại học, số câu đúng chỉ tăng thêm 1. Ở các bài test còn lại, độ chính xác cũng tăng lên từ 3% đến 5% khi sử dụng cơ sở dữ liệu lớn hơn. Từ đó ta có thể rút ra đánh giá khái quát rằng độ chính xác của hệ thống có phần nào phụ thuộc vào bộ ngữ liệu mà ta chọn huấn luyện. Bộ ngữ liệu càng lớn, số lượng các n-grams thu thập được càng lớn và càng đa dạng.

Ở phép đánh giá cuối cùng, ta thấy với phương pháp hiện tại, hệ thống cho độ chính xác cao hơn đối với các dạng câu hỏi về ngữ nghĩa. Các câu hỏi ngữ pháp thường rơi vào các trường hợp chia động từ của câu ở thì thích hợp, các dạng câu if, câu wish. Các câu hỏi về ngữ nghĩa thường thuộc dạng xác định giới từ phù hợp cho động từ đứng trước nó, xác định tính từ, danh từ phù hợp với ngữ cảnh của câu dựa vào nghĩa của câu trước hoặc sau nó. Với phương pháp thống kê, các loại cụm động từ, cụm danh từ, từ ghép hay thành ngữ thông dụng sẽ được lưu vào cơ sở dữ liệu. Cơ sở dữ liệu càng lớn, số lượng cụm từ thống kê được càng lớn thì tỉ lệ trả lời đúng cho câu hỏi dạng ngữ nghĩa cũng tăng theo. Đối với các dạng câu hỏi về ngữ pháp, ví dụ như loại câu chia động từ, câu if, để chọn được đáp án đúng cần phải quan tâm đến các thành phần bổ nghĩa khác. Chính vì vậy phương pháp thống kê cho kết quả không cao.

5.5 Kết luận

Thông qua việc xây dựng hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh dạng điền khuyết, chúng tôi đã nghiệm thu được các kết quả sau:

Chương 5. KHẢO SÁT, ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- Nắm được các kiến thức cơ bản để giải quyết bài toán kiểm tra ngữ pháp trong lĩnh vực xử lý ngôn ngữ tự nhiên.
- Cài đặt thành công phương pháp thống kê để kiểm tra ngữ pháp và ứng dụng cho việc giải bài tập trắc nghiệm điền khuyết tiếng Anh.
- Nâng cao kỹ năng rút trích thông tin từ các nguồn dữ liệu khác nhau (Web, text, XML).
- Hiểu và triển khai mô hình lập trình ứng dụng hướng dịch vụ - Service Oriented Application (SOA).
- Xây dựng được web service bằng công nghệ Windows Communication Foundation, kết nối cơ sở dữ liệu bằng công nghệ Entity Framework.
- Xây dựng ứng dụng di động trên nền tảng Windows Phone theo mô hình 3 lớp.

Tóm lại, mặc dù kết quả ban đầu không cao nhưng hệ thống hỗ trợ giải câu hỏi trắc nghiệm tiếng Anh đã chứng minh được tính ứng dụng của bài toán kiểm tra ngữ pháp vào việc giải các dạng bài tập trắc nghiệm và hứa hẹn khả năng mở rộng hướng phát triển trong tương lai. Bên cạnh đó, với việc xây dựng hệ thống theo mô hình SOA sẽ giúp mở ra khả năng tiếp cận của người dùng đến các ứng dụng khác trong lĩnh vực xử lý ngôn ngữ tự nhiên ngay từ thiết bị di động nhưng vẫn đảm bảo tốt về trải nghiệm người dùng.

5.6 Hướng phát triển

Từ những đánh giá, khảo sát, ta có thể đưa ra các hướng phát triển trong tương lai để hệ thống thêm hoàn thiện, có độ chính xác cao hơn và khả năng ứng dụng vào thực tiễn tốt hơn.

5.6.1 Bổ sung n-grams cho cơ sở dữ liệu

Bên cạnh OANC còn có rất nhiều các bộ ngữ liệu khác, tuy nhiên phần lớn không được cung cấp miễn phí nên việc tiếp cận còn nhiều khó khăn. Trong khi đó, **Google Books n-grams** là cơ sở dữ liệu n-grams do Google xây dựng và cung cấp miễn phí. Từ các tài liệu, sách, tạp chí... Google đã thống kê sẵn các n-grams với kích thước từ 1-grams đến 5-grams và được tổ chức thành các file văn bản. Mỗi n-grams đều được lưu lại với thông tin tần suất xuất hiện của nó trong mỗi năm tính từ năm 1500. Do đó ta có thể tận dụng cơ sở dữ liệu khổng lồ này để nâng cao số lượng n-grams cho hệ thống trả lời câu hỏi trắc nghiệm tiếng Anh.

Cùng với việc thu thập thêm bigrams và trigrams từ Google Books n-grams, chúng tôi quyết định chọn thử nghiệm thu thập thêm các tetragrams (4-grams). Do bộ ngữ liệu này có kích thước rất lớn (có những file zip có kích thước lên đến 10GB) nên chúng tôi sử dụng nền tảng xử lý phân tán Hadoop của Apache để xử lý dữ liệu và đưa vào cơ sở dữ liệu. Tại thời điểm viết bài vẫn đang trong quá trình thu thập nên không thể đưa ra khảo sát đánh giá cụ

thể.

5.6.2 Ứng dụng hệ luật dẫn để cải thiện độ chính xác

Như trong phần đánh giá đã đề cập, những dạng câu hỏi ngữ pháp thường bị chi phối bởi các yếu tố ngữ cảnh, bổ ngữ. Do đó, ta có thể xây dựng các hệ luật dẫn có thể phát hiện các yếu tố phụ thuộc đó nhằm đưa ra đáp án chính xác hơn. Ví dụ, với mẫu câu if, ta có thể xây dựng luật để phát hiện câu điều kiện loại 1, loại 2 hay loại 3 dựa vào thì của các mệnh đề. Hay dùng luật để phát hiện các từ biểu diễn thời gian (yesterday, tomorrow, since, ago...) để xác định đúng thì của câu.

Việc xây dựng hệ luật dẫn thường đòi hỏi khá nhiều thời gian, công sức khảo sát cũng như phương pháp biểu diễn luật hiệu quả.

5.6.3 Cải thiện ứng dụng người dùng, phát triển ứng dụng trên các nền tảng khác

Hiện tại chỉ có ứng dụng trả lời câu hỏi trắc nghiệm tiếng Anh cho người dùng Windows Phone. Tuy nhiên, số lượng người dùng hệ điều hành di động của Microsoft vẫn chưa nhiều, trong khi đó số người sử dụng Android và iOS lại vượt trội hơn. Chính vì vậy, để tăng số lượng người dùng, việc xây dựng ứng dụng trên các nền tảng còn lại là cần thiết. Nhờ xây dựng hệ thống hướng dịch vụ ngay từ ban đầu nên việc mở rộng ra các nền tảng mới rất đơn giản và không tốn quá nhiều chi phí.

Bên cạnh xây dựng các ứng dụng di động cần phát triển thêm ứng dụng web để mở rộng phạm vi đối tượng người dùng.

Tài liệu tham khảo

- [1] Davide Buscaldi, Paolo Rosso, José Manuel Gómez-Soriano, and Emilio Sanchis. Answering questions with an n-gram based passage retrieval engine. *J. Intell. Inf. Syst.*, 34(2):113–134, April 2010. [5](#)
- [2] Ali Çiltik and Tunga Güngör. Time-efficient spam e-mail filtering using n-gram models. *Pattern Recognition Letters*, 29(1):19 – 33, 2008.
- [3] Jesus Gimenez and Lluís Marquez. Svmtool: A general pos tagger generator based on support vector machines, 2004.
- [4] Antje Helfrich and Bradley Music. Design and evaluation of grammar checkers in multiple languages. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, COLING '00, pages 1036–1040, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. [14](#)
- [5] Verena Henrich and Timo Reuter. Lisgrammarchecker: Language independent statistical grammar checking. Master's thesis, Hochschule Darmstadt & Reykjavík University, 2009. [5](#), [6](#), [8](#), [11](#), [12](#)
- [6] Chung-Chi Huang, Mei-Hua Chen, Shih-Ting Huang, and Jason S. Chang. Edit: a broad-coverage grammar checker using pattern grammar. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations*, HLT '11, pages 26–31, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. [11](#)
- [7] Akshat Kumar and Shivashankar B. Nair. An artificial immune system based approach for english grammar checking. In *Proceedings of the 6th international conference on Artificial immune systems*, ICARIS'07, pages 348–357, Berlin, Heidelberg, 2007. Springer-Verlag. [14](#)
- [8] Nay Yee Lin, Khin Mar Soe, and Ni Lar Thein. Developing a chunk-based grammar checker for translated english sentences. In *PACLIC*, pages 245–254, 2011. [11](#), [13](#), [14](#)

- [9] M. Maybury. Natural language processing: System evaluation. In Keith Brown, editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 518 – 523. Elsevier, Oxford, second edition edition, 2006.
- [10] Boryana Miloshevska. *Silverlight for Windows Phone Toolkit in Depth*. windowsphonegeek, 2011.
- [11] Rogelio Nazar and Irene Renau. Google books n-gram corpus used as a grammar checker. In *Proceedings of the Second Workshop on Computational Linguistics and Writing (CLW 2012): Linguistic and Cognitive Aspects of Document Creation and Document Engineering*, EACL 2012, pages 27–34, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. [11](#), [12](#), [14](#)
- [12] Karel Oliva. Techniques for accelerating a grammar-checker. In *Proceedings of the fifth conference on Applied natural language processing*, ANLC '97, pages 155–158, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.
- [13] Jong C. Park, Martha Stone Palmer, and Clay Washburn. An english grammar checker as a writing aid for students of english as a second language. In *ANLP*, pages 24–, 1997. [11](#)
- [14] Charles Petzold. *Programming Windows Phone 7*. Microsoft Press, 2010. [52](#)
- [15] Gerasimos Potamianos and Frederick Jelinek. A study of n-gram and decision tree letter language modeling methods. *Speech Communication*, 24(3):171 – 192, 1998.
- [16] Brian Roark, Murat Saraclar, and Michael Collins. Discriminative n-gram language modeling. *Computer Speech & Language*, 21(2):373 – 392, 2007.
- [17] John Sharp. *Windows Communication Foundation 4 Step by Step*. O'Reilly Media, Inc, 2010. [29](#), [32](#)
- [18] Cornelia Tschichold, Franck Bodmer, Etienne Cornu, Francois Grosjean, Lysiane Grosjean, Natalie Ktibler, Nicolas Lrwy, and Corinne Tschumi. Developing a new grammar checker for english as a second language. In *In Proc. Of 33 rd Annual Meeting of ACL*, pages 189–196, 1997. [11](#)
- [19] Tom White. *Hadoop: The Definitive Guide 3rd edition*. O'Reilly Media, Inc., 3rd edition, 2012. [44](#)
- [20] E.J. Yannakoudakis, I. Tsomokos, and P.J. Hutton. n-grams and their implication to natural language understanding. *Pattern Recognition*, 23(5):509 – 528, 1990.

TÀI LIỆU THAM KHẢO

- [21] Imed Zitouni. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88 – 104, 2007.